TU WIEN Informatics

Extraction of Cyber Threat Intelligence from Raw Log Data

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Max Landauer
Matrikelnummer 01228830

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao.univ.Prof. Dr. Andreas Rauber

Diese Dissertation haben begutachtet:

Tanja Zseby

Christopher Kruegel

Wien, 22. Dezember 2021

Max Landauer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Informatics

Extraction of Cyber Threat Intelligence from Raw Log Data

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Max Landauer
Registration Number 01228830

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.univ.Prof. Dr. Andreas Rauber

The dissertation has been reviewed by:

Tanja Zseby

Christopher Kruegel

Vienna, 22nd December, 2021

Max Landauer

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Max Landauer

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. Dezember 2021

Max Landauer

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Acknowledgements

First and foremost, I thank Prof. Andreas Rauber for supervising this dissertation. His continuous support and guidance over the last years were a major factor in shaping the dissertation to what it is today. Our discussions about new ideas and concepts were always very positive and encouraging, and he consistently managed to point out some seemingly tiny details that eventually turned out to be crucially important for the results of this thesis. I also want to express my gratitude to Florian Skopik, without whom this dissertation would never have come into existence. Florian came up with the overall topic and created the best possible circumstances for me to carry out the research work. His ingenuity was essential to identify all relevant research problems and his ideas contributed to all concepts presented in this thesis. Despite his busy schedule, he was always engaged in the subject of my work and never hesitated to take the time to support me.

I would like to thank all my fellow colleagues from the cyber security team at the Austrian Institute of Technology, with whom it has been a pleasure to work with. I specifically thank everyone who contributed to this thesis: Markus Wurzenberger, Georg Höld, Wolfgang Hotwagner, and Maximilian Frank. Without them, it would not have been possible to achieve all the goals of this thesis.

I sincerely thank my parents, Regina and Walter Landauer, who gave me all the possibilities to pursue my goals. I am more than grateful for their continuous support throughout my whole life. I also thank Anneliese and Hans Steiner for the countless dinner invitations in the last years that kept me alive. I thank everyone in my family for always being able to rely on them. I hope they are happy now that there is finally a doctor in our family. I also want to thank all of my friends for always being there for me.

Last but not least, I thank the Austrian Research Promotion Agency (FFG) for financially supporting this thesis. I don't know about them, but in my opinion the money for the project INDICAETING (868306) was well spent.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Kurzfassung

Heutzutage sind digitale Systeme allgegenwärtig. Dabei geht dieser sich kontinuierlich ausbreitenden Vernetzung trotz vieler Vorteile auch eine große Anzahl von bisher unvorstellbaren Bedrohungen einher. Ein großes Problem sind Cyber-Angriffe, die Schwachstellen von Computersystemen ausnutzen, um diese zu zerstören oder Daten zu stehlen. Betroffene Organisationen haben dabei oft nur wenig Möglichkeiten, um auf solche gezielten Angriffe zeitnah zu reagieren, was zu finanziellen Verlusten und Rufschädigungen führt. Um Angriffen vorzubeugen, werden deshalb sogenannte Intrusion-Detection-Systeme (IDS) zur Systemüberwachung eingesetzt. Signatur-basierte IDSs sind in der Lage für bestimmte Angriffe typische Muster in Logdaten zu lokalisieren, können aber keine unbekannten Angriffe erkennen und erfordern kontinuierliche Aktualisierungen der verwendeten Signaturen. Dazu kommen noch die Probleme, dass Signaturen aufgrund ihrer Einfachheit oft leicht zu umgehen, ungeeignet zur Beschreibung komplexer Angriffe, und schwierig zu erstellen sind. Eine Alternative stellen Anomalie-basierte IDSs dar, die durch maschinelles Lernen auch unbekannte Angriffe erkennen können. Nachteile dieser Methode sind jedoch die vergleichsweise höhere Anfälligkeit für Fehlalarme sowie die Notwendigkeit von Expertenwissen zur Interpretation der Anomalien. In dieser Dissertation wird deshalb ein Ansatz vorgestellt, der die Vorteile beider Methoden kombiniert. Das Ziel ist es, aus Sequenzen von Anomalien sogenannte Meta-Alarme zu generieren, die ähnlich wie Signaturen die Erkennung derselben oder ähnlicher Angriffe auf anderen Systemen ermöglicht. Zu diesem Zweck wird eine Methode zur Aggregation von Alarmen und Anomalien präsentiert, die unabhängig von IDSs, Angriffen, oder Systemen funktioniert. Der Ansatz gruppiert Ereignisse nach Auftrittszeitpunkten und verwendet Ähnlichkeitsmetriken, um Gruppen zu Meta-Alarmen zusammenzufassen. Die Evaluierung des Ansatzes erfolgt insbesondere durch ein öffentlich verfügbares Anomalie-basiertes IDS. Im Rahmen dieser Dissertation wird dieses IDS um ein Modul erweitert, dass die statistische Analyse von Korrelationen in kategorischen Werten von Logdaten ermöglicht. Die Evaluierung dieser Angriffserkennungsmethode erfordert die Verfügbarkeit von Logdaten. In der Dissertation wird daher auch eine Methode zur automatischen Erstellung von Testbeds zur Logdatengenerierung vorgestellt. Durch die modellgetriebene Abstraktion von Testbed-Komponenten ist es dabei möglich, beliebig viele Testbeds mit variabel festgelegten Parametern zu erzeugen. Da somit die Systeminfrastruktur, das Systemverhalten, sowie das Angriffsszenario Variationen unterliegen, sind die resultierenden Logdaten repräsentativ für verschiedene Systemumgebungen und dadurch besser geeignet für Evaluierungen.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Abstract

The omnipresence of digital systems has led to an interconnected economy and society. Unfortunately, the introduction of new technologies in the rapidly expanding global networks has also enabled previously unimaginable threats. Cyber attackers are utilizing advanced tools and techniques to compromise systems and exploit vulnerabilities for the purpose of data exfiltration and destruction. Frequently targeted victims are corporations or organizations that often have no methods in place to detect such targeted attacks in time, resulting in financial and reputational losses. As a consequence, cyber security deploys so-called intrusion detection systems (IDS) to monitor system behavior and disclose suspicious activity. While signature-based IDSs that search for predefined patterns in logs are highly effective, they are unable to detect unknown attacks and rely on manually maintained databases of attack signatures. The main problem with such signatures is that they are often easy to evade and too simple to detect complex attack cases, and that their generation is slow and relies on domain knowledge. Anomaly-based IDSs seem to resolve some of these issues by leveraging machine learning to detect unknown attacks, however, are notorious for high false positive rates and produce anomalies that are difficult to interpret and relate to specific attacks. The idea presented in this dissertation is therefore to combine the advantages of both methods by generating so-called meta-alerts from sequences of anomalies that enable detection of the same or similar attacks on other systems, as achieved by signatures. For this purpose, a new alert aggregation mechanism is proposed that does not rely on any predefined knowledge about the deployed IDSs, observed attacks, or monitored systems. In particular, the method groups anomalies and alerts by their occurrence times and uses similarity metrics to cluster and merge groups into meta-alerts. For evaluation of the approach, anomalies are generated by a publicly available anomaly-based IDS. As part of this dissertation, this IDS is extended by a concept for analyzing categorical values in log data. Thereby, statistical tests are used to recognize changes in value correlations as anomalies. Evaluating the ability to detect attacks requires labeled log data. The dissertation therefore also proposes a method for automatic testbed deployment. In particular, testbeds are instantiated from abstract templates following principles from model-driven engineering. This enables to generate arbitrary numbers of testbeds with dynamically assigned random values for specific testbed parameters, which introduces variations in the infrastructure, normal system behavior, and attack executions. The resulting log datasets are representative for diverse system environments and thus improve evaluations.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Problem Statement & Goals 2
1.2 Contributions 5
1.3 Organization of the Thesis 8

2 Background & Related Work **11**
2.1 Log Datasets 11
2.2 Anomaly Detection 17
2.3 Cyber Threat Intelligence 22
2.4 Alert Aggregation 25

3 Simulation Testbeds **31**
3.1 Testbed Design Methodology 33
3.2 Testbed Models 38
3.3 Validation 46
3.4 Discussion 50
3.5 Summary 52

4 Log Data Generation & Labeling **53**
4.1 Methodology 54
4.2 Scenario 56
4.3 Analysis of Log Datasets 69
4.4 Discussion 83
4.5 Summary 87

5 Log Clustering **89**
5.1 Survey Background 90
5.2 Survey Method 95

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.3 Survey Results . . . 98
5.4 Summary . . . 115

6 Anomaly Detection **117**
6.1 Concept . . . 118
6.2 Approach . . . 121
6.3 Evaluation . . . 127
6.4 Discussion . . . 134
6.5 Summary . . . 134

7 CTI Extraction **137**
7.1 Entities & Operations . . . 139
7.2 Framework . . . 150
7.3 Implementation of the Framework . . . 154
7.4 Evaluation . . . 162
7.5 Summary . . . 175

8 Conclusion **177**

List of Figures **181**

List of Tables **185**

Bibliography **187**

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 1

Introduction

The past few years have seen a strong trend towards digitalization in many different areas. This includes Industry 4.0 leveraging interconnected processes, automation, and decision support [Gho20], Internet of Things such as sensors and other physical objects that communicate with each other over networks [AKBS19], cloud computing where data storage and processing is outsourced to external infrastructure [KKA14], and the omnipresence of smart phones for communication, payment, entertainment, and more. In addition, there has recently been a rapid but extensive shift towards remote work and education as a consequence of the COVID-19 pandemic [BHO+20]. These transitions lead to a situation where computers permeate all layers of economy and society, a concept also known as ubiquitous computing [MC17].

While transitions to new technologies and procedures usually come with benefits in terms of efficiency and convenience, it is undeniable that the increased complexities of systems and a higher level of interconnectedness also led to unprecedented attack vectors that pose a threat to people and organizations. This is clearly visible in recent reports on the cyber security landscape, which indicate that cyber attacks have not only continuously increased in numbers in the last years, but are also more frequently involving sophisticated exploits; a tendency that is expected to continue in the future [ENI21]. In particular, adversarial actors are continuously working on targeted intrusion methods that evade existing detection mechanisms [Cro21]. Unfortunately, this implies that security analysts are most often one step behind the attackers.

When intrusions are successful, attackers usually have multiple days or even weeks until their presence is detected [SGIM21], which gives them enough time to exfiltrate data or damage systems. Since attack attempts themselves can hardly be prevented, it is therefore essential to drastically reduce the time required to recognize system infiltrations. The most common method for automatic attack detection involves monitoring continuously generated log data for known indicators of malicious events [KGVK19]. However, generating cyber threat intelligence (CTI) such as attack signatures is non-trivial as

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

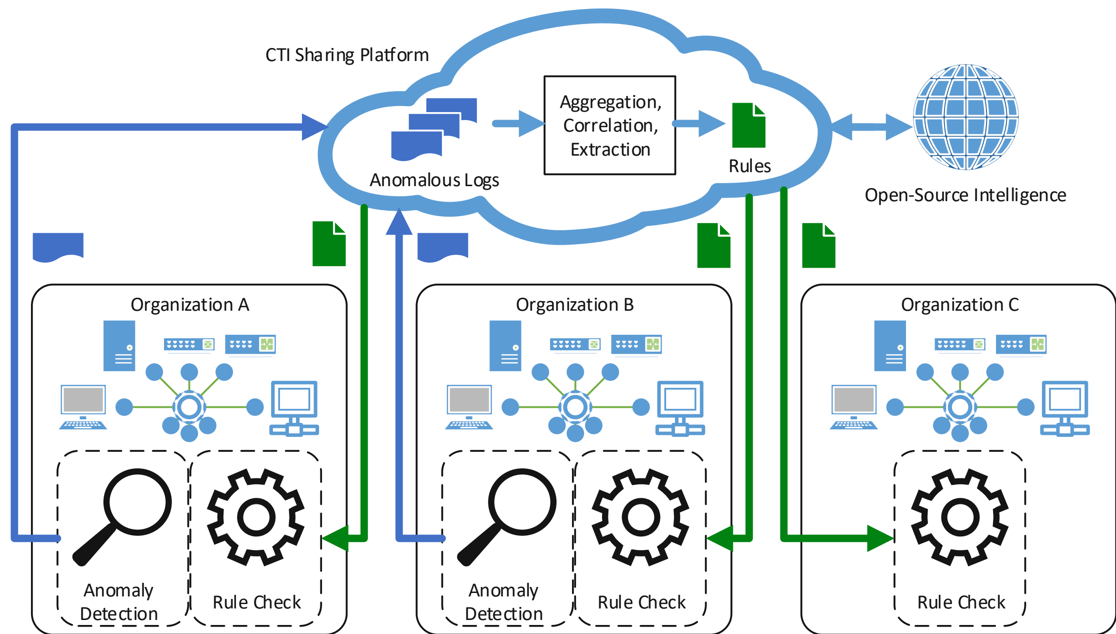


Figure 1.1: Concept for sharing anomalies to enable automatic attack detection and classification across organizations.

it involves manual analysis of attack consequences and careful consolidation of attack indicators into a single detection rule by a domain expert. To ease this situation and enable faster detection, there is therefore a need for an automatic mechanism that detects system activity related to attacks, extracts relevant traces from log data, and transforms it into CTI that enables immediate detection of the same attack on other systems.

Figure 1.1 depicts an exemplary scenario employing such an approach. In particular, organization A and B have similar system infrastructures (i.e., networks involving the same technologies for servers and hosts) and furthermore have IDSs in place that detect unusual system activities as anomalous log events. By publishing their findings on a common CTI sharing platform it is possible to determine whether both organizations are affected by the same attack and use aggregation mechanisms to automatically transform the events into a generic detection rule. All participating organizations profit from the resulting CTI, including organization C that may also be affected by the same attack.

1.1 Problem Statement & Goals

Cyber attacks make it necessary to deploy intrusion detection systems (IDS). These applications monitor computer systems or networks and execute countermeasures or at least notify administrators in case that any malicious activity is recognized. Many approaches thereby focus on network traffic for detection, however, only have limited visibility of attacks due to encryption of network packets. On the other hand, system log

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

data is usually not affected by encryption, but occurs in many different formats, which complicates analysis.

Independent of the ingested data, the most common type of detection involves signature-based IDSs that search for predefined patterns that are known to correlate to specific attacks. Their popularity is due to the fact that they are very effective in detecting known attacks, have low false alarm rates, and are able to process large amounts of logs with high efficiency [KGVK19]. Unfortunately, they are unable to detect unknown attacks for which no signatures exist and also rely on a database of signatures that needs to be maintained manually. In addition, signatures are usually only simple Indicators of Compromise (IoC) such as hashes or IP addresses that are easy to evade for attackers. Accordingly, these IoCs have limited reliability and only enable temporary protection against cyber attacks [LSW+19, Rho14, TR18].

An alternative type of CTI are Tactics, Techniques, and Procedures (TTP). In contrast to IoCs, they are abstract descriptions of attack cases and are thus generally valid. However, due to their abstract nature, TTPs are primarily designed to support manual attack analysis in forensic investigations and thus do not provide any indicators that feed into signature-based IDSs, which prevents automatic attack recognition [HASA+17, Rho14].

Another form of intrusion detection that avoids aforementioned issues altogether is anomaly detection. Other than signature-based IDSs that rely on a predefined set of indicators, anomaly detection leverages methods from machine learning to automatically generate a baseline of normal system behavior and subsequently detect any deviation from that model as anomalous and potentially malicious [CBK09]. The main advantage of anomaly-based IDSs is that they detect unknown attack cases without relying on manually defined detection rules. However, anomaly detection generally suffers from a higher false alarm rate since rare and unusual system activities are frequently reported as anomalies [KGVK19]. Furthermore, as anomalies only relate to specific properties of the data, they usually do not provide any labels for specific attacks, but instead rely on semantic interpretation by analysts. Unfortunately, this is often infeasible as anomalies occur in large numbers, a situation also known as alert flooding [LSWR22].

To address the high quantity and diverse structure of anomalies, aggregation and correlation methods that group alerts into clusters are required. Thereby, correlation of anomalies across systems requires fuzzy matching as the properties of anomalies are subject to variation depending on the system infrastructure where they originate from. Unfortunately, development and evaluation of such approaches is difficult as no benchmark datasets exist that contain anomalies generated in diverse but similar environments. In particular, these datasets need to contain training log data from diverse sources that represent normal system behavior as well as repeated executions of attack scenarios with variations [NK19, SW99].

The aforementioned problems concern several aspects of log data analysis for security applications, including generation of training data, detection techniques used in IDSs, and properties of CTI. The problem statement of this thesis is thus summarized as follows.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

- Existing log datasets do not involve repeated attack executions in diverse system infrastructures. Accordingly, they are not suitable for evaluation of IDSs or alert aggregation approaches due to a lack of variation of both benign and malicious attack consequences in log data.

- Signature-based IDSs are unable to detect unknown attacks. Anomaly detection alleviates this problem, but suffers from high false positive rates and lack of attack labels. This makes it difficult for analysts to aggregate and interpret the findings. In addition, detection within system log data is often limited as the data involves diverse formats and heterogeneous events.

- Neither IoCs nor TTPs enable detection or classification of complex attack cases. While IoCs are too simple and easy to evade, TTPs do not provide detectable indicators or patterns suitable for IDSs.

There is a need to alleviate aforementioned problems. The goal of this dissertation is therefore as follows.

To develop a procedure and an analytic model that enables the automatic or semi-automatic (i.e., including human validation) extraction of Indicators of Compromise (IoC) as well as Tactics, Techniques and Procedures (TTP) from basic unprocessed raw log data with limited knowledge about the infrastructure under investigation.

The problem statement and goal of the dissertation raise several research opportunities and challenges. The following list enumerates the research questions to be answered in the dissertation.

RQ1 What is an appropriate method to enable flexible generation of realistic and labeled log data that facilitates evaluation of CTI extraction approaches?

RQ2 Which methods are appropriate to parse raw and unstructured log data to enable unsupervised event classification, parameter analysis, and incremental anomaly detection?

RQ3 To what extent can a detection method be designed to analyze categorical values in parsed log data for the purpose of identifying anomalous system states with high accuracy?

RQ4 What is an appropriate method to technically describe advanced attacks involving multiple consecutive attack steps as artifacts and complex system behavior patterns in an abstract way to provide reusability on other systems?

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

1.2 Contributions

Major parts of this dissertation were published in conference proceedings, journal papers, magazine articles, and book chapters. The following list enumerates the main contributions of this dissertation and states the most relevant publications.

1. **Automatic generation of testbeds for log data collection and labeling.** There exist only few datasets containing system log data and most of them do not fulfill the requirements for realistic evaluation of anomaly detection and alert aggregation approaches, including a sufficiently long and complex training phase of normal system activity, repeated executions of attack cases with variations, and collection of raw logs from diverse sources. We therefore propose a method for the automatic generation of testbeds using concepts from model-driven engineering to ease deployment and configuration of components. Thereby, the main idea is to specify the infrastructure, normal behavior and attacks as templates that are dynamically instantiated for each testbed [LSW+21]. We also present a methodology that assigns labels to malicious events in alignment with the model-driven nature of our testbeds [LFS+22]. Based on our concepts, we generate and publish a collection of labeled log datasets for evaluation of IDSs and alert aggregation approaches [LSF+].

   - **Landauer M.**, Skopik F., Wurzenberger M., Hotwagner W., Rauber A. (2021): Have It Your Way: Generating Customized Log Data Sets with a Model-driven Simulation Testbed. IEEE Transactions on Reliability, Vol.70, Issue 1, pp. 402-415. IEEE. [LSW+21]
   - **Landauer M.**, Skopik F., Frank M., Hotwagner W., Wurzenberger M., Rauber A. (2022): Maintainable Log Datasets for Evaluation of Intrusion Detection Systems. Under review. [LSF+]
   - **Landauer M.**, Frank M., Skopik F., Hotwagner W., Wurzenberger M., Rauber A. (2022): A Framework for Automatic Labeling of Log Datasets from Model-driven Testbeds for HIDS Evaluation. Proceedings of the Workshop on Secure and Trustworthy Cyber-Physical Systems, forthcoming. ACM. [LFS+22][1]

2. **Anomaly detection techniques for system log data.** Most state-of-the-art anomaly detection techniques analyze features of network traffic. Unfortunately, network traffic is often encrypted in practice and thus detection in the payload is limited. Log data on the other hand is not encrypted and keeps track of almost all events that take place on a system, but is difficult to parse as it occurs in many different formats and often involves heterogeneous events. We therefore survey log clustering approaches and review their abilities to group log data, detect outliers as anomalies, and provide parsers [LSWR20]. Based on these insights, we create

[1] Parts of the concepts and results presented in this paper were first published in a Master's thesis by Maximilian Frank [Fra21].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

parsers for our own log data and then design anomaly-based detectors. In particular, we propose a sophisticated method that employs statistical tests to select categorical variables in log data that are appropriate for correlation analysis [LHW+21].

- **Landauer M.**, Skopik F., Wurzenberger M., Rauber A. (2020): System Log Clustering Approaches for Cyber Security Applications: A Survey. Elsevier Computers & Security Journal, Volume 92. May 2020, pp. 1-17. Elsevier. [LSWR20]
- **Landauer M.**, Höld G., Wurzenberger M., Skopik F., Rauber A. (2021): Iterative Selection of Categorical Variables for Log Data Anomaly Detection. The 26th European Symposium on Research in Computer Security (ESORICS 2021), October 04-08, 2021, virtual. Springer. [LHW+21]
- **Landauer M.**, Skopik F., Wurzenberger M., Hotwagner W., Rauber A. (2020): Visualizing Syscalls using Self-Organizing Maps for System Intrusion Detection. 6th International Conference on Information Systems Security and Privacy (ICISSP 2020), February 25-27, 2020, Valetta, Malta. INSTICC. [LSW+20b]
- Wurzenberger M., **Landauer M.**, Skopik F., Kastner W. (2019): AECID-PG: A Tree-Based Log Parser Generator To Enable Log Analysis. 4th IEEE/IFIP International Workshop on Analytics for Network and Service Management (AnNet 2019) in conjunction with the IFIP/IEEE International Symposium on Integrated Network Management (IM), April 8, 2019, Washington D.C., USA. IEEE. [WLSK19]
- Skopik F., Wurzenberger M., **Landauer M.** (2021): Smart Log Data Analytics: Techniques for Advanced Security Analysis. 208p., 1st edition, ISBN-13: 978-3-030-74449-6, Springer International Publishing. [SWL21b]
- Skopik F., **Landauer M.**, Wurzenberger M. (2021): Skopik F., Landauer M., Wurzenberger M. (2021): Online Log Data Analysis With Efficient Machine Learning: A Review. IEEE Security & Privacy, forthcoming. IEEE. [SLW21]
- Skopik F., Wurzenberger M., **Landauer M.** (2021): The Seven Golden Principles of Effective Anomaly-Based Intrusion Detection. IEEE Security & Privacy, Vol.19, Sept./Oct. 2021, pp. 36-45. IEEE. [SWL21a]
- Wurzenberger M., Höld G., **Landauer M.**, Skopik F., Kastner W. (2020): Creating Character-based Templates for Log Data to Enable Security Event Classification. 15th ACM ASIA Conference on Computer and Communications Security (ACM Asia CCS), October 05-09, 2020, Taipei, Taiwan. ACM. [WHL+20]
- Skopik F., **Landauer M.**, Wurzenberger M. et al. (2020): synERGY: Cross-correlation of operational and contextual data to timely detect and mitigate attacks to cyber-physical systems. Elsevier Journal of Information Security and Applications (JISA), Volume 54, October 2020. Elsevier. [SLW+20]

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- Skopik F., Wurzenberger M., **Landauer M.** (2020): DECEPT: Detecting Cyber-Physical Attacks using Machine Learning on Log Data. ERCIM News, Number 123, October 2020, pp. 33-34. ERCIM - The European Research Consortium for Informatics and Mathematics. [SWL20]

3. **Alert aggregation techniques for extraction of CTI from raw log data.** Existing alert aggregation approaches rely on the presence of certain attributes in alert formats, in particular, IP addresses. Unfortunately, this is mostly applicable to alerts generated by IDSs that analyze network traffic, as system logs do not necessarily involve network information or other attributes required by these alert formats. We therefore present a similarity-based approach for alert aggregation that does not rely on any domain knowledge about the monitored systems or deployed IDSs [LSWR22]. Moreover, we outline a concept on how to integrate such an aggregation approach into a pipeline that automatically extracts CTI from raw log data [LSW+19].

   - **Landauer M.**, Skopik F., Wurzenberger M., Rauber A. (2022): Dealing with Security Alert Flooding: Using Machine Learning for Domain-independent Alert Aggregation. ACM Transactions on Privacy and Security, forthcoming. ACM. [LSWR22]
   - **Landauer M.**, Skopik F., Wurzenberger M., Hotwagner W., Rauber A. (2019): A Framework for Cyber Threat Intelligence Extraction from Raw Log Data. International Workshop on Big Data Analytics for Cyber Threat Hunting (CyberHunt 2019) in conjunction with the IEEE International Conference on Big Data 2019, December 9-12, 2019, Los Angeles, CA, USA. IEEE. [LSW+19]
   - **Landauer M.**, Skopik F. (2019): INDICÆTING - Automatically Detecting, Extracting, and Correlating Cyber Threat Intelligence from Raw Computer Log Data. ERCIM News, Number 116, January 2019, pp. 25-26. ERCIM - The European Research Consortium for Informatics and Mathematics. [LS19]

The following datasets were created and published in course of carrying out this thesis.

1. **AIT-LDSv1.1** [LSW+20a]. The collection of four log datasets was obtained from testbeds generated by the approach presented in [LSW+21]. In brief, the datasets contain logs from webservers running Horde Groupware and OkayCMS. Over the course of six days, normal user behavior was simulated with state machines and several attacks were launched against the server, including scans, brute force login attempts, remote command executions, and exploits of several vulnerabilities. Log data is collected from diverse sources, including audit logs, access logs, authentication logs, exim logs, mail logs, and syslog.
2. **AIT-LDSv2.0** [LSF+21]. The second version of our dataset comprises logs from eight testbeds. In contrast to the AIT-LDSv1.1, the testbed represents a more

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

complex network structure comprising different zones and includes additional components, such as a file share, VPN server, firewall, etc. We also extended the user simulation to utilize new services and generate more complex behavior patterns. We also improved the mechanism for labeling malicious events. The dataset is described in detail in [LSF+].

Finally, several tools were developed and extended as part of the dissertation. The following enumeration provides an overview of these applications.

1. **AMiner** [AMi]. The AMiner is an open-source log-processing pipeline that parses log data and analyzes the parsed values for anomalies. The AMiner is part of the ÆCID (Automatic Event Correlation for Incident Detection) framework [AEC] developed in course of several research projects. The anomaly detection technique for categorical values proposed as part of this thesis [LHW+21] is available in the ÆCID toolbox and may thus be used as a detector in the AMiner pipeline.
2. **Kyoushi Testbed** [Kyo]. All code that is necessary to deploy and setup testbeds for log data generation is available open-source. This also includes the transformation engine that instantiates testbeds from models, the library for simulating normal user activities and attacks, as well as the event labeling framework.
3. **ÆCID Alert Aggregation** [Ale]. We published the algorithms for alert aggregation as open-source code. The repository also contains all training and testing data used to evaluate the approach. Accordingly, all results are reproducible.

1.3 Organization of the Thesis

This section maps the chapters of the thesis to the proposed concept for CTI extraction from raw log data. Figure 1.2 provides an overview of the pipeline and references the chapter number at the corresponding stage. The main idea of the proposed approach is to generate log data containing traces of attacks in controlled environments, cluster the log data to generate parsers, disclose anomalies in the parsed logs, and aggregate the anomalies into meta-alerts that represent specific attack steps. The resulting meta-alerts are then suitable to detect the same attacks on other systems with fuzzy matching.

The thesis is structured as follows. Chapter 2 reviews the state-of-the-art for all research areas relevant for this thesis and also provides background information on CTI. The presented literature was used to disclose existing research problems and identify opportunities for new developments. The chapter is divided into sections for several stages of the pipeline, in particular, log data testbeds and datasets, anomaly detection, CTI, and alert aggregation.

Chapter 3 outlines an automatic approach for testbed generation using model-driven concepts. The main idea is to define testbeds as abstract models that are dynamically

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

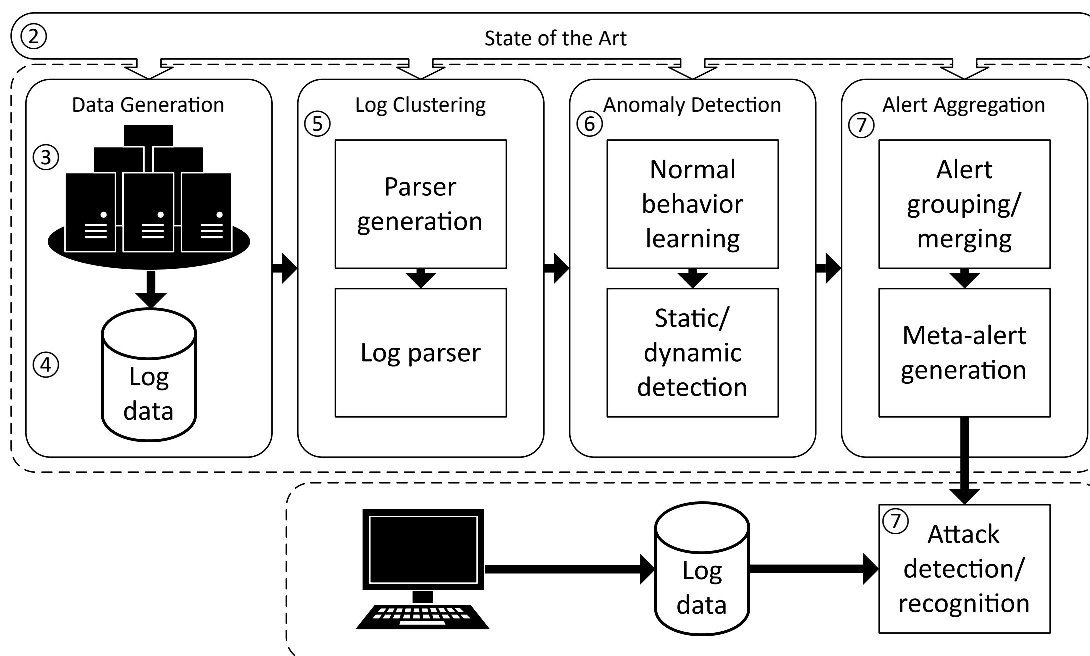


Figure 1.2: Pipeline of the proposed approach for CTI extraction from raw log data.

instantiated with random variables to introduce variations to the system infrastructure, normal behavior simulations, and attack executions. Chapter 4 extends this concept with a rule-based labeling mechanism. In particular, the labeling rules are automatically filled out by facts derived from the testbeds so that no manual modifications are required when executing them across different testbed instances. The chapter also provides some insights into the generated datasets and compares them with log data collected from real system infrastructures.

Chapter 5 provides a survey on log clustering approaches in the security domain and groups them based on a predefined list of requirements. Thereby, the generation of parsers appears as one of the most relevant goal of clustering. Chapter 6 uses such a log parser to extract categorical values from log events and carry out anomaly detection. The approach is designed to identify correlating values through a series of filtering stages in a training phase and then disclose anomalies as changes of these correlations with the aid of statistical tests.

Violations of correlation rules may also appear as part of normal system behavior and attacks often cause that multiple detectors raise anomalies. Accordingly, anomalies and alerts are frequent in actively used systems and it is non-trivial to relate them to specific attacks. Chapter 7 therefore proposes an alert aggregation algorithm that groups and merges alerts that are considered similar. When sufficiently many alert groups that correspond to the same attack are merged from different environments, the resulting merged group only involves those properties that are present in the majority of all attack

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

observations. This means that the resulting merged group is representative for a specific attack execution, i.e., a so-called meta-alert. Applying the same similarity metrics used for meta-alert generation allows to detect and classify attacks that occur on other systems. Accordingly, meta-alerts are considered CTI as they provide complex patterns similar to TTPs but also include detectable IoCs.

Chapter 8 concludes the thesis. The chapter provides answers to the research questions and states recommendations for future work.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 2

Background & Related Work

This chapter provides background information on the concepts discussed in this thesis and furthermore describes and compares relevant approaches from the state-of-the-art. As the scope of this thesis spans across several research fields that deserve separate analysis, the remainder of the chapter is divided into the following sections.

- **Log datasets.** This section first outlines requirements on log datasets available for evaluation of IDSs and alert aggregation approaches and then presents some common datasets that are publicly available.
- **Anomaly detection.** This section covers some background of anomaly detection, including types and methodologies of IDSs. Some well-known anomaly detection techniques are described, in particular, with a focus on value correlations.
- **Cyber threat intelligence.** This section explains the terms Cyber Threat Intelligence (CTI), Indicator of Compromise (IoC), and Tactics, Techniques, and Procedures (TTP). In addition, some approaches that extract CTI from various data sources are presented.
- **Alert aggregation.** This section states several requirements for alert aggregation techniques and checks whether these requirements are met by state-of-the-art methods.

2.1 Log Datasets

This section reviews and compares existing log datasets used in security research. Major parts of this section have been published in [LSF$^+$]. Due to the large need for datasets for IDS evaluation, several attempts to generate benchmark datasets were made in the past. However, most of these datasets are created with specific use-cases in mind and are

thus not generally applicable. To compare these datasets on a common scheme, we first describe a set of requirements that are relevant for intrusion detection datasets and then discuss the fulfillment of these aspects for several state-of-the-art datasets.

Requirements

Recording log datasets in testbeds or real environments is not straightforward; it is a task that requires careful planning, since the quality and usefulness of the resulting data strongly relies on several decisions made by the analyst. We gathered a list of requirements by reviewing design principles that were followed by authors of existing datasets. In the following, we summarize our findings.

(1) **Use-case**. To ensure relevance and authenticity of the dataset, it is necessary to design the overall network layout and technical infrastructure of the system where log data is recorded in the context of a specific scenario. This also includes services available on the involved machines [LSW+21]. Clearly specifying the scope of the simulation also helps to define the limitations of the dataset.

(2) **Synthetic data generation**. Datasets collected from real-world system environments are sometimes considered superior to synthetically generated data due to the fact that they are per definition realistic, while simulations only try to replicate their characteristics [HHS+17]. However, real datasets have the strong disadvantage that it is infeasible to differentiate normal from anomalous or malicious logs with complete certainty, since the root causes of some actions are unknown to the analysts [SW99]. Obviously, synthetic dataset generation implies that scripts that replicate normal behavior on an appropriate level of detail are prepared beforehand. This particularly concerns models for user activities that normally occur on the system, which can be very diverse and thus non-trivial to formalize. On the plus side, modeling the normal behavior effectively enables to steer the parameters of the simulation to generate data that is representative for different levels of detection complexity [SW99]. Therefore, we argue that synthetically generated log datasets are the best option for IDS evaluations.

(3) **Attacks**. As part of a realistic evaluation of IDSs, it is necessary to select recent and relevant attack scenarios that are suitable for the system environment at hand [RWG+17,LSW+21]. Otherwise, outdated attack cases may not yield representative intrusion detection evaluation results that are comparable to that of more modern attacks.

(4) **System logs**. When IDSs are applied in productive systems, they are usually able to analyze logs in raw and unaltered form. Accordingly, log datasets for evaluation of IDSs should also provide logs that are not processed in any way [MFCMC+18]. Fortunately, synthetic datasets recorded in simulations are usually less critical when it comes to privacy, since no humans are involved and thus anonymization

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

of personal user data that possibly occurs in the logs is not required. This also concerns sensitive contents of files that may appear in the logs and should thus be simulated with collections of predefined dummy files [UHH+21]. Another important aspect is to configure the logging framework in a realistic way that fits the use case. For this, analysts must decide where to log and what to log [ZHF+15]. In particular, anomaly-based IDSs require logs corresponding to normal system behavior to learn a baseline for detection, meaning that logging levels should be set to *info* or even *debug* rather than *error* or *warning*. Moreover, it is beneficial to log performance metrics such as CPU or memory data, because they are also adequate inputs for IDSs [KAW11].

(5) **Network traffic**. Beside system logs that are the main input of host-based IDSs, network traffic is a widely used data source for network-based IDSs. Accordingly, datasets should also include packet captures to enable evaluation of network-based IDSs and hybrid IDSs that make use of both system logs and network traffic [LLLT13].

(6) **Periodicity**. Productive system environments naturally exhibit periodic behavior, for example, cron jobs are scheduled for execution in fixed intervals and events originating from human activities follow daily and weekly patterns of work shifts. Self-learning IDSs are able to integrate these cycles in their models to detect contextual anomalies, i.e., events that are considered anomalous due to their time of occurrence [CBK09]. It is therefore essential to expand the duration of the simulation to cover several of these cycles [MFCMC+18].

(7) **Labels**. Ground truth tables that unambiguously assign labels to all events are needed to compute evaluation metrics such as detection accuracy or false alarm rates [RWG+17]. Accordingly, it is essential to provide a comprehensible methodology for creating correct ground truth tables for IDS evaluation.

(8) **Documentation**. Datasets should be published with detailed descriptions of all relevant aspects of the data creation. Otherwise, it is not possible for others to fully understand all artifacts present in the data, which could possibly lead to incorrect assumptions and invalidate evaluation results [MFCMC+18].

(9) **Repetitions**. For anomaly-based IDSs that only learn from normal behavior and then classify test data either as normal or anomalous, it is sufficient to only have artifacts of a single attack execution in the data. However, for attack classification it is necessary that attacks are at least present in training and test datasets, and possibly validation datasets. Accordingly, attacks should be launched multiple times by repeating the simulation. In addition, research on alert aggregation urgently requires useful datasets, especially for system logs analyzed by host-based IDSs [NDP18]. Thereby, clustering-based aggregation methods require that the same attacks are carried out multiple times to form groups.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

(10) **Variations**. Approaches for both attack classification and alert aggregation should be challenged by introducing variations in attack executions [SW99]. Moreover, evaluation results have a higher robustness when they are based on multiple attack executions that cover a spectrum of possible attack variations [LSW+21]. This behavior could be realized by dynamically changing attack parameters in each simulation run.

(11) **Reproducibility**. Technologies that constitute the simulation are continuously updated. To avoid that datasets become outdated, it should be possible to repeat simulations at any given time [TSB08, UHH+21]. This also allows to reuse existing assets and only change certain parts of the simulation, e.g., keep the infrastructure and user simulation, but include another attack vector. It is therefore beneficial to publish all code used to carry out the simulation alongside the resulting datasets.

Literature Analysis

The previous section outlines a set of requirements that should be fulfilled by datasets to enable evaluation of intrusion detection systems. We gathered several datasets that are commonly used in scientific evaluations and analyzed whether they fulfill our requirements. Table 2.1 shows a complete list of all datasets and our findings, where ✓indicates that the datasets fulfill the respective requirement, ∼ indicates partial fulfillment, and no symbol means that the requirement is not fulfilled. In the following, we discuss our findings and relevant properties of the datasets in detail.

One of the earliest log datasets that became widely used in intrusion detection is the KDD Cup 1999 dataset [SFL+99]. The logs were collected during a simulation of several intrusions in a military network. Other than many modern datasets, the authors made sure to label all events with the respective attack types and furthermore repeat and vary the attacks to yield different probability distributions in the training, validation, and test datasets. These properties make this dataset especially attractive for evaluating machine learning techniques. Even today it is still widely used in scientific publications, although the dataset has been repeatedly criticized for being outdated, too simple, and not reproducible due to the fact that closed-source tools were used for traffic generation [TSB08].

As a consequence of these criticisms, Creech et al. generated ADFA-LD [CH13] and ADFA-WD, two datasets containing sequences of system calls on a Linux and Windows host respectively. For the generation of the dataset, the authors simulated normal activities such as web browsing and file editing and launched several attacks, such as brute-force logins and exploits for webshell uploads. Unfortunately, the system calls are stripped from all contextual variables such as timestamps, parameters, and return values, and are thus not representative for real data [ČG18]. Moreover, the dataset is criticized for only including a single host, not generalizing well for other systems, as well as a lack of documentation detailing how the dataset is collected and what services are installed [AHH20, MFCMC+18]. The AWSCTD [ČG18] aims to resolve at least one of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 2.1: Fulfillment of requirements for existing datasets

		Requirement									
Dataset	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**	**(11)**
ADFA-LD [CH13]	Linux OS		✓	∼			✓	✓	✓		
ADFA-WD [Cre14]	Windows OS		✓	∼			✓	✓	✓		
IoT-DDoS [AHH20]	Internet of Things	✓	✓		✓	∼	✓				
AWSCTD [ČG18]	Windows OS		✓	✓	✓	∼		✓			
CIDD [KB12]	Cloud Systems		✓	✓	✓	✓	✓	✓			
CIDDS [RWG+17]	Enterprise IT	✓	✓		✓	✓	✓				
LID-DS [GRKG19]	Linux OS	✓	✓	✓			✓	✓	✓		
VAST Challenge 2011 [GCH+11]	Enterprise IT		✓	✓	✓	✓	✓	✓			
KDD Cup 1999 [SFL+99]	Military IT				✓	✓	✓	✓	✓	✓	
Loghub [HZHL20]	Supercomputer and OS			✓		✓		✓			
NGIDS DS [HHS+17]	Enterprise IT	✓	✓		✓		✓				
CICIDS 2017 [SLG18]	Enterprise IT	✓	✓		✓	✓	✓	✓			
Skopik et al. [SSFF14]	Enterprise IT	✓	✓	✓	✓			✓			
SOCBED dataset [UHH+21]	Enterprise IT	✓	✓	✓	∼			✓	✓	∼	✓
UGR'16 [MFCMC+18]	Enterprise IT		✓		✓	✓	✓	✓	✓		
HDFS [XHF+09]	Supercomputer			✓		∼	✓				
AIT-LDSv1.1 [LSW+21]	Enterprise IT	✓	✓	✓		✓	∼	✓	✓	✓	
AIT-LDSv2.0 [LSF+]	Enterprise IT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

these issues by recording Windows system calls without removing any parameters and further extend the set of launched attacks. However, the authors also consider only a single host and not a full network.

Another dataset based on Linux system calls is LID-DS [GRKG19]. While the authors explain the attack scenarios in great detail, there is only little information on the simulation of normal system behavior. They carry out all attacks multiple times and collect the logs from hundreds of runs that last around 30 seconds each. CIDD [KB12] provides logs specifically for masquerade attacks. One of the noteworthy aspects of this data is that the authors manage to label all events by correlating network and system logs and mapping them to attack tables specifying the expected times, IP addresses, and user names related to attacks. Moreover, the users generating normal activity in the dataset are categorized into normal, advanced, administrators, programmers, and secretary users.

One of the few datasets that also include system logs other than system calls is from the VAST Challenge 2011 [GCH+11]. In particular, the dataset comprises firewall logs, IDS alerts, syslogs and network packet captures. Among the attacks launched against the simulated system are security scans, denial-of-service attacks, and remote desktop connections as consequences of a social-engineering attack. The authors also provide a document describing the solutions to the challenge that depict a ground truth of malicious events. The dataset presented alongside the open-source testbed called SOCBED [UHH+21] contains system logs from a network of Windows and Linux hosts. While the authors did not collect network traffic for this dataset, they state that it is simple to extend their testbed accordingly and repeat the experiments. In addition, the authors discuss variations in log data, however, only with respect to circumstantial factors such as system performance and not purposefully incorporated variations as accomplished by our model-driven approach. Skopik et al. [SSFF14] also collect network traffic as well as access and application logs on a testbed where simulated users click around on a mail platform. Contrary to most existing papers that present new datasets, they configure their user simulations based on behavior of real users and also validate their data by comparison of accessed resources. Other datasets comprising system and application logs from various services are provided in Loghub [HZHL20]. The main problem with these datasets is that they mainly involve traces related to failures rather than cyber attacks.

While system log datasets are most often collected from single hosts, whole networks comprising several hosts are usually deployed to generate network traffic datasets. For generating CIDDS [RWG+17], the authors recreated a virtual company with network components that are commonly used in enterprise IT, e.g., Windows and Linux hosts as well as file shares and web servers, and place them in separate subnets for managements, office, and developers. Their user simulations are based on state machines to generate complex behavior patterns instead of repeated sequences and their models also respect working hours and breaks. Moreover, their network is also connected to the Internet to mix the simulated traffic with real connections and possibly attacks. To generate the UGR'16 [MFCMC+18], the authors also use a combination of real user behavior

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

traffic and simulated attack traffic. Thereby, the authors specifically pay attention to the cyclic behavior of communication logs that originates from daily or weekly usage patterns. Moreover, their attacks are generated with random starting times.

The authors of CICIDS 2017 [SLG18] follow a different approach as make use of a profiler that analyzes real communication in a network and then arbitrarily generates data following these patterns. They recorded the network traffic while launching several attacks, among which are denial-of-service attacks, vulnerability exploits, and a botnet. Similarly, a network traffic generation appliance was also used to generate NGIDS DS [HHS+17]. Other than these datasets, the IoT-DDoS [AHH20] specifically focuses on a scenario that simulates Internet of Things in a network.

In Chap. 3 we present the AIT-LDSv1.1 [LSW+21], a system log dataset collected from a webserver hosting a content management system and groupware. Other than most existing approaches for dataset generation, the chapter describes a model-driven strategy for automatic testbed deployment to generate multiple datasets with variations of attack executions. We recognize several shortcomings of the dataset: First, beside some machines running user simulations, the network is relatively simple as it only consists of a single webserver. Second, the simulation focuses on system log data and thus no network traffic is captured. Third, labeling of malicious events is not reliable since it relies on similarity-based matching, which may lead to incorrectly unlabeled lines in case that variations lead to new or dissimilar events [LFS+22]. Finally, only the resulting data is publicly available, but the scripts for deploying the testbed and running the simulation are not accessible. As a consequence of these shortcomings, we propose the AIT-LDSv2.0 [LSF+] in Chap. 4. In comparison to our previous testbed for log data generation, we increased the network complexity, collected logs from all components of the network (e.g., the firewall), extended the simulation of normal behavior, improved the strategy for event labeling, and published all code for deploying the testbed along with the generated dataset. As visible in Table 2.1, our new dataset meets all requirements stated in Sect. 2.1. We discuss the fulfillment of these requirements in detail in Sect. 4.4.1.

2.2 Anomaly Detection

This section discusses anomaly detection techniques applicable for log datasets as described in the previous section. First, the characteristics of different anomaly detection methods are stated. Then, some state-of-the-art detection techniques are explained.

2.2.1 Overview

IDSs monitor continuously generated log data with the purpose of detecting and reporting suspicious or malicious system activities. Thereby, it is common to differentiate between two commonly used data sources: network traffic and system log data. In the following, the two types of IDSs are described [KGVK19, CBK09].

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

- **Network-based IDSs** monitor network traffic, such as packet captures or netflows. Accordingly, they monitor several systems that are connected to a particular network and specifically detect external malicious activities or attackers moving laterally through the network. One of the main problems is that deep packet inspection is usually impossible due to encryption, which limits the visibility of attacks. In addition, network-based IDSs are unable to detect any malicious activity taking place on hosts rather than the network, such as insider threats.

- **Host-based IDSs** are agents that are installed directly on the hosts they monitor. They typically process system log data, including audit logs, logs from the operating system, firewall logs, application logs, etc. As system log data keeps track of almost all events that take place in a system and is usually not encrypted, host-based IDSs are able to perform detection on a high level of granularity and also enable detection of insider threats.

Independent of the type of analyzed data, there are generally three different methodologies that determine the design and application of IDSs. The main characteristics of each approach are stated in the following [SM+07].

- **Signature-based detection** makes use of predefined patterns to recognize known attacks. These signatures are usually simple strings, e.g., IP addresses or hash values. While this type of detection is fast and effective, it is unable to detect unknown attacks for which no signatures exist and relies on signature databases which need to be manually maintained.

- **Anomaly detection** learns models for normal system behavior and detects divergences from these models as potentially malicious activities, which enables the detection of previously unknown attacks. Since system behavior is usually subject to change over time, a dynamic baseline that allows incremental updates is required to avoid that models become outdated or inaccurate. On the other hand, dynamic models (and training phases in general) are susceptible to adversarial machine learning, where attackers purposefully introduce certain artifacts that are incorrectly learned as normal and thereby prevent subsequent detection of similar activities that are related to attacks. Anomaly detection also suffers from a high false positive rate as benign but rare events are sometimes detected as anomalies, especially when they did not occur during training.

- **Stateful protocol analysis** relies on universal protocol profiles that are usually developed by vendors. While these profiles are relatively precise and thus enable fine-grained detection, they are often highly complex and thus difficult to create. In addition, they require manual updating whenever implementations change.

For the context of this thesis, anomaly detection is the most relevant technique. When it comes to self-learning systems as applied for anomaly detection, it is common to

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

differentiate between supervised, semi-supervised, and unsupervised approaches. In the following, each learning technique is briefly explained [KGVK19, CBK09].

- **Supervised anomaly detection** requires the presence of labels for all training log events. Unfortunately, it is usually infeasible in practice to manually generate these labels due to the sheer volume of log data and the fact that not all activities that produce events are known by the analyst. Another problem for many machine learning techniques is that labels are usually highly imbalanced, as events related to attacks are far fewer than events related to benign activities.
- **Semi-supervised anomaly detection** only requires that instances of normal events are labeled as such. In practice, this usually means that there exists a training dataset that contains only benign activities and is thus free of anomalies, and dedicated test datasets containing both normal and malicious events. Since no labels for attacks are required, these approaches are more widely applicable than supervised detection techniques.
- **Unsupervised anomaly detection** do not require any labels at all. These methods usually base on the assumption that anomalous instances are far fewer than benign instances, which enables detection techniques such as clustering or outlier detection. Since no training data is required, unsupervised anomaly detection is the most widely applicable technique.

As supervised anomaly detection techniques are hardly applicable in practice, the focus of this thesis lies on semi-supervised and unsupervised approaches. Anomaly detection has been an active field of research for many years and many such approaches have been proposed in scientific literature. One of the first well-known approaches for anomaly detection in log data was proposed by Forrest et al. [FHSL96], who use a sliding window of a fixed size to learn a set of allowed sequences of system call operations during a training phase. After processing a sufficient amount of logs, training is stopped and all newly observed sequences that are not included in the trained model are detected as anomalies. Even today, anomaly detection in sequences of log data is still widely pursued, however, makes use of neural networks. Du et al. propose DeepLog [DLZS17], a detection tool that uses a Long Short-Term Memory (LSTM) neural network to recognize unusual sequences of log events. One of the most interesting aspects of their approach is that it allows incremental updating of the model, which is difficult to achieve when applying neural networks, but essential for semi-supervised learning in continuously generated log data. Zhang et al. present a similar approach called LogRobust [ZXL+19] that is especially designed to handle noise in log data, in particular, events that vary due to changes of logging statements. Huang et al. [HLF+20] extend this approach to also include the parameters of events when detecting unusual log sequences.

The diverse characteristics of different log types and formats are an explanation for the various approaches for anomaly detection that have been proposed in the past. In

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

addition, authors often make different assumptions on the data they analyze. Other than aforementioned approaches on sequence detection, Juvonen et al. [JSH15] do not rely on any knowledge about the data or even require parsing; instead, they split log lines into n-grams and then apply projection methods (e.g., principal component analysis) to detect outliers. Other approaches that operate on virtually any type of log data often involve string metrics. For example, Wurzenberger et al. [WSL+17] propose an incremental similarity-based clustering approach that discloses dissimilar lines as outliers that may relate to suspicious or otherwise interesting events.

In contrast, some anomaly detection approaches require that the syntax of analyzed log data is known so that it is possible to extract and individually analyze specific values. In particular, Kruegel et al. [KV03] analyze the presence, lengths, character distributions, orders, and structures of values extracted from HTTP traffic for detection. Other approaches make use of parsed log data to generate event count matrices, i.e., sets of vectors that describe how often known events occur within a predefined time window. He et al. [HZHL16] state that these vectors are suitable for grouping events into normal and abnormal clusters using similarity thresholds, outlier detection using principal component analysis, and extraction of invariants, i.e., linear relationships such as dependencies between events.

Chapter 5 provides a survey that reviews many additional detection approaches using static clustering that focuses on single events or dynamic clustering where collections of events are considered (e.g., event sequences or frequencies). For this reason, the following section focuses on detection techniques for correlations of categorical values, as they are relevant for the detection approach presented in Chap. 6.

2.2.2 Anomaly Detection in Categorical Variables

Several anomaly detection techniques were stated in the previous section. This section reviews approaches that are specifically developed for anomaly detection in categorical variables. Note that major parts of this section have been published in [LHW+21].

Research on association mining between categorical variables in database transactions has been ongoing for many years. One of the main issues prevalent in this field is the immense search space arising from the many possible combinations of variables and values [TH19]. Accordingly, approaches such as the well-known Apriori algorithm [AS+94] are usually designed for efficient searching and pruning.

To enable outlier or anomaly detection in categorical data, it is usually necessary to adjust or extend association mining algorithms. For example, Narita and Kitagawa [NK08] propose techniques to detect records that fail to occur in expected associations and to compute outlier scores that are also suitable for speeding up the search. Khalili and Sami [KS15] show that the Apriori algorithm is suitable to be used for intrusion detection, in particular, by identifying critical states of industrial systems with sensor outputs as variables. One of the downsides of algorithms based on frequent itemset mining is that they require multiple passes over the data, which prevents online processing. Djenouri et

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

al. [DBFV18] therefore propose a single-pass technique with improved parameter selection and use pruning to limit the search space to itemsets that cover the largest amount of events.

The problem with such approaches based on frequent itemset mining is that they omit infrequent values, because they are not interesting for the associations. Anomalies are then considered as infrequent combinations of otherwise frequent values [TH19]. However, infrequent values are important for anomaly detection, as long as they occur consistently with their associated values. Accordingly, Das and Schneider [DS07] replace rare values with placeholders and use conditional probabilities to disclose associations.

Distance-based techniques are commonly used for anomaly detection in numeric data, however, it is non-trivial to compute distances between categorical values. Eiras-Franco et al. [EFMRGB+19] solve this problem by encoding categories as binary vectors to apply maximum likelihood analysis. Similarly, one-hot encoding is also used by Moustafa and Slay [MS16], who measure the association strength between variables using the Pearson correlation coefficient as well as Information Gain. Ren et al. [RWZH09] support anomaly detection on data streams by computing cluster references on chunks of data, where a distance function based on value equality is used.

A different strategy to tackle the lack of a distance metric and large event space is pursued by Chen et al. [CTS+16], who embed the data in a latent space and mine associations between pairs of variables, which include user IDs, IP addresses, and URLs. Similarly, Pande and Ahuja [PA17] use an embedding method based on word2vec for anomaly classification in HTTP logs. Alternatively, Ienco et al. [IPM16] measure the similarity between value co-occurrences by applying distance metrics on their conditional occurrence probabilities. The advantage of this method is that it enables anomaly score computation for ranking. Conditional probabilities are also used by Tuor et al. [TKH+17], who show that neural networks are suitable for anomaly detection in categorical user data. The downside of these approaches is that they suffer from lower explainability than frequent itemset methods, where variable associations are more intuitive.

Most aforementioned approaches rely on the assumption that their data involves only categorical variables or that these variables have been manually pre-selected. However, log files involve various data types, including discrete, continuous, static, and unique variables. Gupta and Kulariya [GK16] therefore use a Chi-squared test to select variables with sufficiently distinct value co-occurrences before comparing regression, support vector machines, naive bayes, and decision trees for anomaly detection.

Chapter 6 presents an approach for anomaly detection in categorical values of log events. Similar to some of the aforementioned approaches, our detection method analyzes chunks of log data rather than individual lines for this purpose. However, other than existing approaches, we employ a sequence of constraints to limit the search space and then make use of statistical tests to disclose anomalies.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.3 Cyber Threat Intelligence

This section explains the term CTI and states advantages and disadvantages of relevant types of CTI. In addition, some techniques for CTI extraction are discussed. Major parts of this section have been published in [LSW+19].

2.3.1 Term

The term Cyber Threat Intelligence (CTI) is used highly ambiguously throughout all kinds of literature. In particular, it is unclear at what point any available information on cyber threats is regarded intelligent rather than just data.

Chismon and Ruks [CR15] define CTI through the process of detecting and subsequently analyzing previously unknown threats with the aim of understanding and mitigating risks. Zhu et al. [ZD18] state that unlike automatically collected data, generating CTI encompasses manual threat analysis and reasoning by domain experts. McMillan [McM13] provides a definition that involves evidence-based knowledge and context information on mechanisms, indicators, implications, and actionable advice about existing or emerging threats. Dalziel et al. [Dal14] state that CTI must be refined, analyzed, and processed in order to be relevant, actionable, and valuable.

The consensus of these definitions is that security-related data needs to undergo a process of advanced analysis and enrichment to provide usable insights into cyber threats and be regarded as actionable CTI. The term actionable is thereby used just as ambiguously as the term threat intelligence itself. Dalziel et al. [Dal14] denote CTI as actionable if it is specific enough to enable decision-making and response to present threats. Tounsi et al. [TR18] point out that outdated CTI loses its actionability, but mention that fast sharing of CTI is not sufficient to prevent targeted attacks. They also discuss the relevance of standardized CTI formats to ensure data quality and enable automated analysis. Popular CTI formats are STIX [Bar12], IODEF [IOD], OpenIOC [Opea], and CAPEC [Bar06].

We conclude that actionability means that no additional analyses are necessary to utilize available CTI; however, the specific requirements on actionable CTI depend on the desired use-case, such as detection, analysis, or containment.

Chismon and Ruks [CR15] separate CTI into four subtypes: (i) technical, i.e., low-level Indicators of Compromise (IoC) with limited validity, (ii) tactical, i.e., low-level information on Tactics, Techniques and Procedures (TTP) with longer validity, (iii) operational, i.e., high-level details on imminent threats, and (iv) strategic threat intelligence, i.e., high-level reports on organizational risks. Due to their relevance to this thesis, we discuss IoCs and TTPs in the following sections.

2.3.2 Indicators of Compromise

IoCs are typically described as artifacts which presences provide concrete evidence that system security was breached with high confidence. The well-known STIX [Bar12] format

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

defines indicators as "patterns that allow detection of suspicious or malicious cyber activity". Patterns thereby include IP addresses, email addresses, domain names, and computed indicators such as hash values. They are highly specific and thus IDSs usually report only few false alarms.

The actionability of IoCs is debatable and depends on the use-case at hand. On the one side, Tounsi et al. [TR18] state that IoCs are immediately actionable, because they can be automatically fed into Intrusion Detection Systems (IDS) once they become available. On the other hand, detection based on predefined IoCs is more reactive than proactive, i.e., detecting an IoC usually means that the system is already compromised. Moreover, Rhoades [Rho14] argues that IoCs are too simple to identify complex malicious system activities. Tounsi et al. [TR18] even state that a key failing of CTI is that it is relatively simple for adversaries to ensure that attacks generate no artifacts that match pre-existing IoCs.

Another important aspect pointed out by Chismon and Ruks [CR15] is that IoCs from different CTI feeds yield small overlaps. Their explanation is twofold: First, it is easy to vary attack parameters such as IPs. Second, existing CTI is not of any intelligent value. Either way, these observations discredit the actionability of IoCs. Finally, one more problem with IoCs is that they are usually consumed by IDS without their context of occurrence, i.e., an IoC is either observed in the data or not [Rho14]. To alleviate these issues, a more abstract way of describing threats is required. In the following section, TTPs are reviewed as a method to provide such information.

2.3.3 Tactics, Techniques and Procedures

While IoCs are detective in nature, Tactics, Techniques and Procedures (TTP) provide abstract and descriptive characterizations of threats, typically in human-readable form [Bar12]. The main purpose of these descriptions is to detail the modus operandi, i.e., actions that attackers carry out on affected systems, and how these actions are carried out, e.g., exploits of certain vulnerabilities.

Among the advantages of TTPs over IoCs is that they are valid for longer time spans and that their abstract descriptions increase the difficulty of evasion by attackers. The reason for both effects is that it is relatively difficult for adversaries to discover completely new ways of executing attacks in comparison to the low efforts of changing artifacts such as IP addresses [TR18]. This is also represented in the so-called "Pyramid of Pain" [Bia14] that places TTPs as the most valuable type of CTI on top and IoCs at the bottom.

The main problem is that it usually takes extensive manual work and domain knowledge to generate TTPs on an adequate level of abstraction. Furthermore, the currently wide-spread human-readable descriptions of TTPs impede their usage for automatic detection [HASA+17, Rho14].

For example, consider the entry "Embedding Scripts within Scripts" (CAPEC-19) in the CAPEC database [CAPa]. The attack is detailed on the "Standard" abstraction level and

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

contains an extensive description of the typical attack execution flow. However, based on the available texts, it is not possible to manually or automatically extract indicators that support attack detection for particular systems.

Enriching existing manually defined TTPs with measurable indicators that support automatic detection of attacks or attack steps could improve this situation. Our research efforts are therefore directed towards bringing IoCs and TTPs closer together by combining and mapping IoCs to TTPs to yield more intelligent indicators.

2.3.4 CTI Extraction

Alert aggregation and the detection of multi-step attacks are well-researched problems in cyber security. Thereby, most approaches aggregate, correlate, and connect alerts by the similarity of certain features. For example, the approach proposed by Julisch [Jul03] generalizes alerts received from IDSs by aggregating their attributes using hierarchies. Valdes and Skinner [VS01] use features such as IP and port lists, user and sensor IDs, and time to compute the similarity for probabilistic alert correlation. Qiao et al. [QZLS12] also take IP, time, and the type of alerts into account for similarity computation and then use LCS to extract the attack patterns. Pei et al. [PGS+16] generate a graph based on the presence of certain attributes and then perform graph community extraction to derive attack patterns. Almost all such approaches rely on the assumptions that attack steps are detectable by traditional IDSs, i.e., that predefined signatures for detection exist, and that alerts are available in well-structured formats that facilitate feature-based similarity computation. In contrast to these methods, the approach proposed in Chap. 7 focuses on unknown anomalies detected in raw log data and incorporates every attribute available in the log lines as well as detector information for similarity computation.

Mapping security events to existing TTPs is able to enrich human-readable attack descriptions with detectable events. Scarabeo et al. [SFK15] use methods from text mining to map IDS alerts to attack descriptions provided by CAPEC [Bar06]. Navarro et al. [N+17] derive context and patterns from log events to generate a complex attack model suitable for matching with TTPs from threat databases.

Other approaches automatically generate CTI from sources other than log files. Husari et al. [HASA+17] use machine learning to extract threat actions from human-readable CTI reports. They then map the sequences of actions to known attack patterns and output them in STIX [Bar12] format. Zhu and Dumitras [ZD18] propose an algorithm that automatically analyzes online security articles and generates detectable patterns that consist of combinations of IoCs. Among the drawbacks of these methods is that they rely on the availability of manually written threat reports, which do not necessarily provide a comprehensive view on the affected systems. In addition, creating these reports is time-consuming and thus threat information is not immediately available after incidents. We argue that log data captures the system in more detail and enables real-time CTI generation and detection.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

2.4 Alert Aggregation

The previous section outlined some techniques for the generation of CTI. One possibility to automatically generate CTI is the aggregation of alerts into higher-level meta-alerts. This section therefore reviews and compares existing methods for alert aggregation. Major parts of this section have been published in [LSWR22].

2.4.1 Requirements

Alert aggregation has been an active field of research for many years. This section therefore reviews the state-of-the-art of alert aggregation in scientific literature. We first outline a number of requirements and then evaluate how each of these requirements is met by existing approaches.

We recognize that the development of alert aggregation techniques is usually motivated by specific problems at hand. Accordingly, existing approaches base on different assumptions regarding available data, acceptable accuracy, tolerable manual effort, application scenarios, etc. To compare existing approaches with respect to the issues outlined in the beginning of this chapter, we define the following list of requirements that should be addressed by domain-independent alert aggregation techniques.

(1) **Automatic.** Manually crafting attack scenarios is time-consuming and subject to human errors [NDP18]. Therefore, unsupervised methods should be employed that enable the generation of patterns and meta-alerts relating to unknown attacks without manual interference.

(2) **Grouping.** Attacks should be represented by more than a single alert. This grouping is usually based on timing (T), common attributes (A), or a combination of both (C).

(3) **Format-independent.** Alerts occur in diverse formats [NDP18]. Methods should utilize all available information and not require specific attributes, such as IP addresses.

(4) **Incremental.** IDSs generate alerts in streams. Alert aggregation methods should therefore be designed to derive attack scenarios and classify alerts in incremental operation.

(5) **Meta-alerts.** Aggregated alerts should be expressed by human-understandable meta-alerts that also enable automatic detection [EO11]. Thereby, generated patterns are usually based on single events (E), sequences (S), or a combination (C) of both.

In the next section, we present the state-of-the-art of alert aggregation techniques. Thereby, we determine which of the aforementioned requirements are met by the reviewed approaches.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 2.2: Fulfillment of requirements for existing alert aggregation approaches

	Requirement				
Approach	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
Al-Mamory et al. [AMZ09]	$\checkmark$		$\sim$	$\checkmark$	E
Alserhani et al. [Als16]		A			S
Alhaj et al. [ASZ$^+$16]	$\checkmark$		$\checkmark$		
Bateni et al. [BBG13]		A			S
Cuppens et al. [CM02]		A	$\checkmark$		E
De Alvarenga et al. [DABJM$^+$18]	$\checkmark$	A			S
Haas et al. [HWF19]	$\checkmark$	A			S
Hofmann et al. [HS09]	$\checkmark$		$\checkmark$	$\checkmark$	E
Husák et al. [HČLV17]	$\checkmark$	A		$\checkmark$	E
Husák et al. [HK19]	$\checkmark$	A		$\checkmark$	S
Julisch [Jul03]	$\checkmark$		$\sim$		E
Liang et al. [LCWX16]	$\checkmark$		$\sim$		
Long et al. [LSS06]	$\checkmark$		$\checkmark$		
Man et al. [MYWX12]	$\checkmark$		$\sim$		E
Moskal et al. [MYK18]	$\checkmark$	A			S
Navarro et al. [NDP16]		T	$\checkmark$		S
Ning et al. [NCR02]		A	$\checkmark$		S
Patton et al. [PBS$^+$11]	$\checkmark$		$\checkmark$		
Pei et al. [PGS$^+$16]	$\checkmark$	A			S
Ramaki et al. [RAA15]	$\checkmark$	A			C
Ren et al. [RSG10]		C	$\sim$		S
Saad et al. [ST12]	$\checkmark$		$\sim$		E
Sadoddin et al. [SG09]	$\checkmark$	A		$\checkmark$	S
Shittu et al. [SHGH$^+$15]		A			S
Spathoulas et al. [SK13]	$\checkmark$	C		$\checkmark$	S
Sun et al. [SG$^+$20]	$\checkmark$		$\checkmark$	$\checkmark$	
Vaarandi et al. [VP10]	$\checkmark$		$\sim$		E
Valdes et al. [VS01]		A	$\checkmark$		S
Valeur et al. [VVKK04]	$\checkmark$	A		$\checkmark$	E
Wang et al. [WC16]	$\checkmark$	A			S
Zheng et al. [ZXH11]	$\checkmark$		$\sim$	$\checkmark$	E
Landauer et al. [LSW$^+$19]	$\checkmark$	C	$\sim$	$\checkmark$	C
Landauer et al. [LSWR22]	$\checkmark$	C	$\checkmark$	$\checkmark$	C

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.4.2 Literature Analysis

This section provides an in-depth analysis of existing alert aggregation techniques with respect to the requirements stated in the previous section. Table 2.2 shows the fulfillment of these requirements, where ✓ or a letter corresponding to the requirement mark a sufficient fulfillment, $\sim$ marks partial fulfillment, and no symbol means that the requirement is not addressed in the respective paper.

Alert aggregation techniques are divided into similarity-based methods that cluster alerts based on common attributes, sequential-based methods that model causal relationships between alerts as conditions, and case-based methods that employ predefined expert rules for correlation [SMFDV13]. Clearly, case-based methods rely on human attack specification and therefore do not fulfill requirement (1). Sequential-based methods on the other hand are more flexible regarding the detection of attacks, for example, the LAMBDA framework proposed by Cuppens et al. [CM02] models attack scenarios using pre- and post-conditions based on alert properties. Ning et al. [NCR02] present a similar mechanism with a higher focus on representing attack scenarios as graphs. Alserhani et al. [Als16] build upon these ideas and create reduced graphs that act as meta-alerts.

Unfortunately, sequential-based methods have limited ability to extract unknown attack scenarios [NDP18] and thus do not fulfill requirement (1). The same applies to approaches that rely on supervised learning, such as algorithms for artificial immune systems used by Bateni et al. [BBG13], or ant colony optimization through reinforcement learning used by Navarro et al. [NDP16]. Our literature analysis shows that only similarity-based methods are capable of fulfilling requirement (1) [NDP18].

Some approaches are designed for aggregation of single alerts only and therefore do not fulfill requirement (2). These approaches mainly address alert filtering and the generation of alert templates. For example, Julisch [Jul03] propose one of the first well-known approaches for alert aggregation, which computes similarities between alert attributes based on generalization hierarchies for specific attribute types, e.g., IP addresses and ports. The approach by Al-Mamory et al. [AMZ09] builds upon these concepts and uses generalization hierarchies to compute alert cluster representatives that are then used for comparison. One of the issues with these hierarchies is that they are defined manually and therefore require mapping to specific attributes. This is solved by Long et al. [LSS06], who propose an XML-based similarity metric for alerts in Intrusion Detection Message Exchange Format (IDMEF) [IDM]. Similarly, Zheng et al. [ZXH11] present type-dependent similarity metrics for pre-selected attributes and use the mean, mode, and set unions of specific attribute values to generate meta-alerts.

To enable attack classification rather than alert filtering, the context of alerts needs to be included in the analysis. One possibility to achieve this is to arrange alerts in sequences by their occurrence time [NDP16]. This is based on the idea that alerts relating to the same root cause likely occur in short time intervals [SMFDV13]. The most common approach however is to link alerts with coinciding values in particular attributes, e.g., De Alvarenga et al. [DABJM$^+$18] and Husák et al. [HČLV17] group alerts by source

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

or destination IP, Shittu et al. [SHGH+15] use fuzzy combinations of IP address parts and ports, Moskal et al. [MYK18] use sequences of alerts with corresponding destination IP and attack category, and Pei et al. [PGS+16] use a total of 29 comparisons of IP addresses, ports, process IDs, host names, etc.

Some approaches consider both timing and attribute correspondence as relevant for alert group formation. In particular, Spathoulas et al. [SK13] group alerts by their occurrence time and use a weighted similarity metric specifically designed to compare time differences and parts of IP addresses. The purpose of their approach is to display attacks as clusters on IP ranges for visual analysis. Ren et al. [RSG10] also create groups for alerts that occur in close temporal proximity and have coinciding attribute values, which are represented as generalization hierarchies. In our earlier work [LSW+19], we first determine alert types by attribute similarity and then create sequences of these alert types based on their interarrival times. The problem with this strategy is that alert types are generated without considering their context of occurrence. Our approach proposed in Chap. 7 alleviates this problem by introducing similarity metrics for groups of alerts that allow to cluster only those groups that relate to the same root cause. Since both timing and attribute similarity are leveraged, the presented approach implements a combined strategy for requirement (2) as we will discuss in Sect. 7.4.11.

Reviewing existing literature with respect to requirement (3) shows that many approaches require that alerts are available in IDMEF format [IDM] or involve at least attributes for source and destination IP addresses, ports, and type. For example, Husák et al. [HK19] propose the AIDA framework, which implements an IDMEF alert processing pipeline that removes duplicate alerts, groups the remaining alerts by source IP, and learns common alert sequence patterns. Haas et al. [HWF19] propose to derive graphs from alerts that represent communication between hosts and allow similarity computation. Even though their approach generates so-called motif graphs that represent abstract attacks and thus do not contain IP addresses and ports, their approach requires IP information for graph generation and thus focuses on alerts from network-based IDSs. Approaches with partial fulfillment of requirement (3) employ generalization hierarchies [Jul03] or similarity functions [ST12, ZXH11] for specific attribute values that are generally valid, but require manual mapping to attributes.

A different approach is proposed by Hoffmann et al. [HS09], who assume statistical distributions for values of arbitrary attributes and therefore fulfill requirement (3). Patton et al. [PBS+11] convert raw text of IDS alerts into vector space models to apply hierarchical clustering. Alhaj et al. [ASZ+16] select features from all alert attributes based on their respective information gain entropy. Sun et al. [SG+20] propose a generalized attribute weighting scheme based on rough sets. Our approach also makes use of all alert attributes and is not restricted to specific formats. However, none of the reviewed techniques include similarity metrics for groups of such alerts, which is solved by our approach.

Incremental clustering as described by requirement (4) is essential for the application in real-world scenarios that involve continuously generated alerts. Sadoddin et al. [SG09]

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

propose one of the few approaches specifically designed for incremental alert processing. In particular, they mine frequent patterns from graphs representing correlated alerts. In addition, Husák et al. [HK19] and Landauer et al. [LSW+19] implement their concepts as pipelines for continuous alert processing. Most approaches however rely on hierarchical clustering [DABJM+18, PBS+11, ASZ+16] or other techniques that do not support incremental processing, or require computation of a correlation matrix prior to alert aggregation [WC16, RSG10]. The main issues are that such algorithms only support offline analysis and require repeated training phases that involve manual supervision.

The final requirement (5) concerns the generation of meta-alerts. Most commonly, meta-alerts are represented as graphs of event sequences, for example, Wang et al. [WC16] create a graph with transition probabilities for correlated alert types and Haas et al. [HWF19] generate graphs that represent abstract communication patterns. Other approaches generate patterns for single alerts, for example, Saad et al. [ST12] use a similarity metric to iteratively refine similar alerts by repeated aggregation, where attributes are replaced by common concepts defined in the generalization hierarchies. Vaarandi et al. [VP10] use frequent itemset mining to obtain patterns of static and variable alert attributes. Valdes et al. [VS01] fuse alerts to meta-alerts by creating supersets of values for shared attributes. Valeur et al. [VVKK04] arrange their meta-alerts in hierarchical structures and apply breadth-first search when merging new alerts. Similarly, we create abstract alert objects in Landauer et al. [LSW+19] and extract sequences of their occurrences. The approach proposed in Chap. 7 also combines alert and event sequence information for meta-alert generation, but ensures to merge only those alerts that occur in a specific sequence position to improve precision of the resulting meta-alerts.

Overall, most of the existing approaches only focus on particular aspects of alert aggregation, e.g., focus on individual alerts rather than groups, rely on domain-specific input in the correlation procedure, or impose strict requirements such as a specific format, in particular, IDMEF [IDM]. Accordingly, none of the approaches fulfill all of our outlined requirements on a domain-independent alert aggregation technique sufficiently. Chapter 7 will therefore propose an approach that addresses aforementioned issues with existing approaches and fulfills requirements (1)-(5).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 3

Simulation Testbeds

The literature research presented in Sect. 2.1 shows that no publicly available datasets are suitable for evaluation of IDSs and alert aggregation approaches. This chapter therefore presents an approach that enables the generation of such datasets in simulation testbeds. Major parts of this chapter have been published in [LSW+21].

The ability to measure and compare the performances between IDSs in a representative way is essential for improving their algorithms and providing new research directions [TSB08]. However, many IDSs are designed and configured for deployment in particular environments and focus on the detection of specific types of cyber attacks. Accordingly, objective IDS benchmarking for selection and deployment in real world applications is non-trivial [WSSS16]. For this reason, research groups have developed simulation testbeds that resemble real networks and allow IDS deployment as well as attack execution in controlled environments [SSFF14, SLG18]. Researchers then publish the network traffic or log datasets collected during their simulation runs to enable others to recreate their results or apply their own methods on the data. Some datasets then become standards for evaluation for some time, however, will at some point be regarded as outdated or criticized for particular aspects, e.g., focus on network traffic rather than log data [ČG18], missing documentation of installed services [MFCMC+18], too simple or unknown simulations of system behavior [TSB08], lack of multi-step attack vectors [NDP18] including long-term advanced persistent threats [SHSK20], missing reproducibility [UHH+21], or too narrow focus that impedes generalization [MFCMC+18]. Eventually, this will encourage other research groups build new testbeds with updated technologies that are relevant for their own use-cases.

Simulation testbeds are essential to validate, evaluate, and compare the capabilities of IDSs. Thereby, testbeds offer analysts environments that yield unbiased results for their use-cases and the real world. Otherwise, it is impossible to reliably assess whether the IDS under test performs with similar efficiency and effectiveness when deployed in productive operation. Furthermore, flawed tests may lead to misconfigurations of IDSs and thus

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 3.1: Testbed development phases

Phase	Involved tasks	Remedies and support
Concept	Define use-cases and goals	-
Design	Component selection	Reuse of code and settings
Deployment	Parameterization and instantiation of components	Tools for automatic system deployment and setup
Utilization	Testing and data collection	Scripts enable automation
Adaptation	Variation of parameters and configuration settings	Automatic selection from predefined ranges or lists

limit their abilities to detect certain attacks. In order to ensure that such requirements are met, high efforts should be spent on preparing and designing the testbed setup.

Setting up testbeds for particular use-cases is usually time-consuming. The most tedious tasks involve adjustments for tests carried out under different conditions as well as follow-up modifications for related use-cases [GdVFM08]. The main problem is that analysts are stuck with rigid testbeds that are set up a single time by domain experts that could not predict the requirements that became necessary after setup. Such testbeds are difficult to maintain or modify for a number of reasons:

- Manual work is required to change the testbed in hindsight, e.g., increase the number of network components.
- Modifications of otherwise identical components have to be repeated multiple times.
- It is necessary to check and update all components individually to ensure that up-to-date versions are used.
- Resetting the testbed to a "clean" state is often necessary to remove artifacts that influence subsequent simulations. However, this undoes purposefully inserted changes and thus complicates iterative testbed development.

Another problem is that most existing testbeds are relatively static, because their configuration, e.g., the selection of a user behavior profile from a predefined list, relies on manual input and domain knowledge. This impedes fast instantiation of different testbeds with variable configurations. In addition, such parameters usually cover only basic application settings of the testbed environment, but are not extensive enough to fine-tune parameters with less influence, e.g., particular aspects of a specific system behavior profile, and not powerful enough to change the overall testbed setup, e.g., upgrade components to newer versions. However, the possibility to obtain multiple testbeds with variations would be highly beneficial for IDS evaluation, because more available data representing different technical environments would enable generation of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

separate training, validation, and test datasets, improve robustness of evaluation results, and support validation of approaches that derive reusable attack attributes and patterns for detection of specific attacks across different systems [NK19, LSW+19].

There is thus a need for a methodology that addresses the aforementioned problems and eases testbed setup and development. Table 3.1 gives an overview of the phases typically encountered during testbed development and states ideas for automation of involved tasks. To integrate these strategies in the development procedure, we propose to leverage techniques from model-driven development. In particular, this dissertation presents an approach that makes use of abstract and testbed-independent models for the testbed infrastructure, system behavior, and attack scenarios, and uses a transformation engine to automatically generate all scripts necessary to setup the infrastructure, configure all components, install services, simulate normal system behavior, and start the attacks. Thereby, our approach is able to generate multiple testbeds instances at once, each with particular characteristics, and produce log datasets that cover variations occurring in different environments. Note that this does not imply that data is generated using model-driven techniques; instead, we propose a model-driven approach for the instantiation of testbeds that are useful for the production of security data.

We implemented a proof-of-concept based on our proposed model-driven methodology where an automatized pipeline allows us to generate arbitrary numbers of testbeds running in parallel. We designed a common real world use-case, i.e., users that access a mail platform and a web store, and launched two attacks that make use of recently discovered exploits. We use this setup to generate four testbed instances with variations and collect their log data, which is published online [LSW+20a]. We summarize the contributions of this chapter as follows:

- A novel model-driven concept for automatically instantiating arbitrary numbers of parameterized testbeds,
- adhering to a set of design principles,
- for the generation of new network and log datasets.

The remainder of this chapter is structured as follows. In Sect. 3.1 we first propose a list of design principles and then introduce our approach for automatic testbed generation using model-driven techniques. Section 3.2 contains concrete design decisions regarding testbed infrastructure, simulation of normal behavior, and attacks. We validate our approach in Sect. 3.3 and discuss applications and limitations of our approach in Sect. 3.4. Finally, Sect. 3.5 summarizes the chapter.

3.1 Testbed Design Methodology

In this section we introduce a methodology for model-driven testbed development. For this, we state design principles that act as requirements for subsequent design decisions.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.1 Design Principles

Most of the issues with existing log datasets generated in testbeds are attributable to shortcomings of the system infrastructures and environments where the data was collected. Accordingly, it is necessary to align the design process of the testbed with requirements on the data to be generated. We therefore pursue a number of design principles that form the basis of our testbed generation methodology. In the following, we briefly discuss each of the principles.

Authenticity

Log data should be collected within realistic scenarios to ensure a representative evaluation of the capabilities of IDSs. Thereby, several aspects must be considered: First, all involved components, e.g., servers or clients, have to be selected and arranged within a network that is representative for a well-defined use-case, e.g., a small enterprise. This includes network complexity, i.e., diversity of involved components, as well as scale, i.e., total number of components. Second, components must act and react like their real counterparts. This includes automatic behavior, e.g., scheduled tasks, as well as user behavior that may be erratic, unpredictable, and dependent on user roles. Third, attacks carried out on the testbed should be related to recently discovered vulnerabilities to ensure that the detection capabilities are not measured on outdated exploits that possibly have lost relevance in modern infrastructures. Accordingly, all services should be set up with fully patched and up-to-date software. Moreover, the attacks should affect common technologies in order to be relevant for a large number of people and organizations. Fourth, the collection of the log data and network traffic has to take place in a realistic manner. This means that only commonly available log sources should be used and that logging should be configured on a level that is adequate for the use-case.

Flexibility

Setting up a testbed encompasses manual time-consuming work [GdVFM08]. It is therefore economically reasonable to design a testbed that is flexible in the sense that it supports adjustments and extensions and enable iterative development. There are mainly three dimensions of modifying the testbed: First, enlarge or shrink the scale of the network by adding or removing components. Thereby, we suggest to initially create a number of predefined components that act as building blocks that can be arbitrarily duplicated and set into relation with each other. Second, the configurations of these components, including all installed services and their versions, are subject to modification. Third, it should be possible to change the dynamic behavior and interactions between the components, e.g., the types of services accessed by clients.

Reproducibility

The ability to reproduce the generated log dataset requires that it is possible to reset the testbed to a past state. This is particularly useful when the effects of an attack on

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

modified versions of the testbed are subject of investigation, for example, comparisons of patched and non-patched services. It is important to note that it is usually impossible to guarantee that the reproduced dataset is identical to the original dataset, but rather only conform in their main characteristics, such as the overall user behavior. This is due to the fact that it is difficult to avoid that latencies during communication of components as well as arbitrarily occurring events or failures result in non-deterministic behavior. In order to ensure reproducibility, it is necessary to isolate the testbed from all external sources that may have unpredictable influence on the outcome of the simulation and are not under control of the analyst, for example, publicly accessible connection over the Internet may cause unknown and potentially malicious behavior manifesting itself in the logs, making subsequent evaluations on the captured data less reliable.

Another important aspect of reproducibility is that only freely available or open-source services are used within the testbed. The reason for this is that the use of commercial products or services that are not publicly available may prevent others from rebuilding the same system.

Availability

To enable IDS benchmarking and comparison of detection capabilities with other approaches, it is important to make generated datasets publicly available. In addition, the dataset has to be accompanied with appropriate documentation, including the overall purpose of the dataset, the infrastructure setup, and a description of the normal and attacker behavior. If such a documentation is missing, it is difficult for others to understand certain artifacts in the data, interpret evaluation results, or reproduce the dataset.

Utilizability

The availability of a dataset alone is not sufficient to enable evaluation of IDSs. In order to obtain comparable evaluation results, a ground truth that defines the malicious behavior in a quantifiable way is needed. Thereby, several levels of labeling the data are possible. The most superficial approach is to label all log events generated during time intervals of attacks as malicious. Note that all other events can be considered benign, because the testbed is a simulation that runs isolated from a productive system that may be affected by unknown processes or attacks. While this form of labeling is easy to accomplish since it is possible to derive anomalous time windows from attack scenario descriptions, it has the disadvantage of also labeling normal events that occur during attacks as anomalous. However, since most IDSs report individual events as anomalies, a more in-depth evaluation is enabled by labeling only events that actually correspond to malicious behavior as anomalous. Even better are labels that differentiate between different types of attacks or attack steps, enabling in-depth evaluation of anomaly detection systems that support attack classification or focus on multi-step attacks.

Anomaly detection systems or other self-learning approaches additionally require that the generated log data covers a sufficiently large duration of the normal behavior in

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

order to be adequately utilized [MFCMC+18]. In particular, the data has to span over multiple cycles of normal behavior, i.e., all repeating processes should be at least once fully present in the data. Incomplete training sets may lead to misclassifications, e.g., false alarms, during evaluation. In addition, log data should always be published in raw format, because any modification such as anonymization or pseudonymization possibly distort evaluation results [SLG18].

3.1.2 Model-driven Testbed Setup Methodology

The usual setup process of a testbed that adheres to the outlined design principles involves time-consuming and non-trivial work. In particular, ensuring flexibility of the testbed, i.e., enabling arbitrary changes of the size of the represented network while at the same time allowing the user to steer component configurations, is technically difficult and involves tedious tasks, such as repeatedly setting up or modifying similar components in slightly different environments. This procedure becomes especially nerve-racking when settings of the system configuration have to match simulated user behavior or are dependent on the type of attack [SSFF14].

We suggest to use techniques from model-driven engineering (MDE) [Sch06] to alleviate these issues. MDE is a methodology that aims at simplifying software development by providing programmers a framework to design solutions on a higher level of abstraction, thereby allowing them to focus on the actual problem at hand independent of technical details and complex implementations on specific platforms. This results in applicable models that support development on multiple different platforms. MDE further makes use of transformation engines that automatically process platform-independent models and generate code for specific platforms.

For our proposed approach, we adopt these concepts from MDE and apply them for testbeds rather than for software platforms. Figure 3.1 shows an overview of the layers of abstraction and workflow that we use for testbed design and automated testbed deployment. Our model-driven approach thereby differentiates between (i) the technical infrastructure, (ii) the normal system behavior, and (iii) the modeled attack.

The top of the figure depicts pre-selected relevant aspects of the real world that is simulated in the testbed. In particular, we seek for commonly available infrastructures that are frequently subject to attacks, such as servers that are accessible over a network. We also look for frequently installed packages and examine the settings of the logging services. Given a real infrastructure, it is also possible to monitor the exhibited behavior and derive relevant characteristics of normal system usage, such as usage distributions over a period of time. Finally, attacks are either observed on the real infrastructure or exist in documented form in online threat databases such as metasploit [Met].

We then define testbed-independent models (TIM). Regarding the infrastructure, this implies declarations of the setup routines for all involved components and services without specifying any concrete parameters. For example, we define how a component is connected to the network, but do not allocate IP addresses, assign names, or specify the number of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

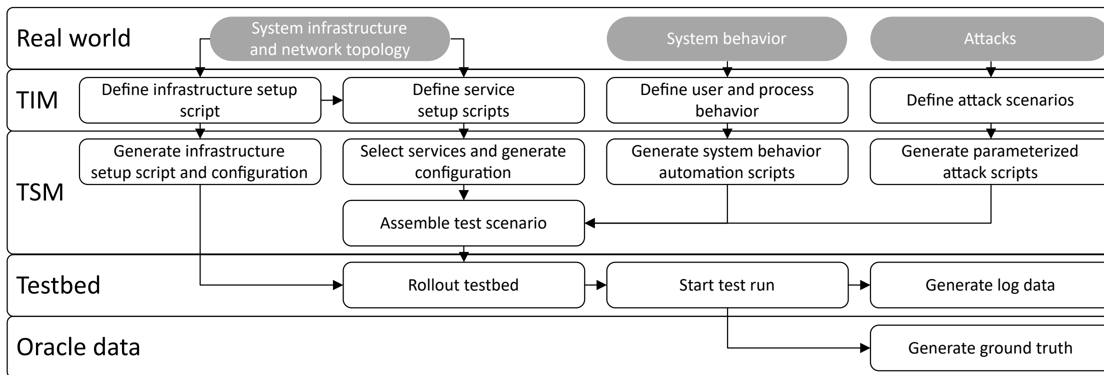


Figure 3.1: Model-driven testbed generation approach. Testbed-independent models (TIM) are derived from real scenarios and transformed to testbed-specific models (TSM) that instantiate the testbed for labeled network and log data generation.

users, but only the type and range of these parameters. Similarly, we design a model of the system behavior as a state machine without fixed transition probabilities between its states, and a model for the attack scenario that consists of the basic steps that are necessary for carrying out the attack. All TIMs function as templates, i.e., they are scripts that represent specific routines, but are configurable through consciously placed parameters throughout the code.

Our transformation engine that generates testbed-specific models (TSM) processes the templates and inserts all parameters to produce executable code. The parameters are thereby selected randomly based on their type specified in the TIM. For example, the number of simulated users is selected from a predefined range, their names and passwords are picked from predefined lists, and IP addresses are automatically assigned from a pool. For the user behavior, we specify a number of profiles with ranges for transition frequencies that the transformation engine translates into probabilities. Regarding the attack scenario, optional parameters of individual steps as well as their order and delays are randomly selected. Note that modeling may be based either on attacks or vulnerabilities, i.e., an attack model scenario may focus on a single malicious action or involve several vulnerability exploits and diverse attack vectors.

Since transformation of TIMs to TSMs is fully automatic, it is possible to generate arbitrary amounts of TSMs at the same time, where each TSM exhibits variations depending on the settings for random selection. For each TSM, we first run the infrastructure setup scripts to build the virtual machines and set up the network of the testbed instance. We then gather, allocate, and run all generated scripts for component setup, user simulation, and attack execution, on the respective machines. After completion, we use another script to copy all logs from the virtual machines and label them according to the outcomes of the simulation. In MDE terminology, such labeled data are usually referred to as oracle data [LPC+13].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

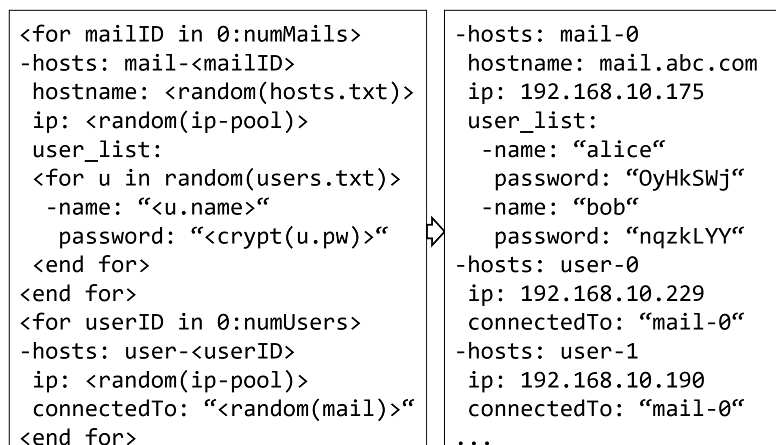


Figure 3.2: Simplified sample transformation of TIM (left) to TSM (right) for infrastructure setup.

3.2 Testbed Models

The previous section outlines the idea of generating testbeds using a model-driven approach. In this section, we discuss selected design aspects of our implementation in more detail.

3.2.1 System Infrastructure

To evaluate our model-driven concept for generating testbeds, we were aiming to create a simulation of a system infrastructure that is common in many organizations. After reviewing usage statistics of well-known technologies, we decided to model an Apache web server hosting a mail platform and content management system (CMS) that are accessed and used by an arbitrary number of users. In particular, we selected Horde Groupware [Hor] and a webshop provided by OkayCMS [Oka], because both platforms are available open-source and have recently been affected by vulnerabilities. Each generated testbed should consist of one web server with a database and a variable number of connected host machines, each representing one or more users.

We designed TIMs in YAML syntax for setup of a web server and a user host machine. Figure 3.2 shows a simplified and shortened version of such a template on the left side, where “mail” refers to the web server and “user” to the user host machine. A transformation engine is able to process such a TIM and generate the TSM on the right side of the figure that acts as a configuration file for the setup procedure. The engine thereby executes the code within the arrow brackets specified in the TIM to fill the gaps of the template with parameters that are subject to change in every testbed instance. For example, the server hostname is randomly selected from a predefined list of names, IP addresses are automatically assigned during setup, and accounts for a random set of users are created.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Before running the transformation engine, it is necessary to specify the total amount of web servers and user host machines to be generated from the TIM. These numbers are critical, since there is usually a limited amount of computational resources available for the virtual machines. The transformation engine then allocates the host machines randomly to web servers (parameter "connectedTo" in Fig. 3.2) in accordance with predefined values for the minimum and maximum number of host machines per web server.

There are two main advantages of designing the infrastructure on this abstract level. First, it is simple to generate large numbers of different testbeds that run in parallel. It is thereby easy to steer the degree of variation by adjusting the predefined ranges in the TIM. Second, changes that affect all components of a particular type only have to be carried out once in the TIM, since these modifications will automatically propagate to all TSMs when running the transformation engine again.

3.2.2 Normal System Behavior

The purpose of the testbed is to generate log data for evaluating attack detection tools. However, executing malicious actions on an idle system makes their detection relatively easy, since almost all generated logs are likely to be related to the attack. This scenario is not authentic, because web servers in the real world are almost always actively used. Furthermore, IDSs based on anomaly detection usually rely on a training phase that represents normal and anomaly-free behavior in order to disclose deviations from the learned patterns.

We therefore simulate normal system behavior by modeling typical user accesses. For this, we created a state machine that covers all relevant functions of both the Horde Webmail and OkayCMS platforms using the well-known web automation framework Selenium [Sel]. Figure 3.3 shows a graphical overview of all subpages and activities that are covered by the state machine and users are thus able to visit. On Horde Webmail, the users are capable of changing their preferences, writing mails to other users and responding to received mails, and creating and deleting entries in the calendar, notebook, list of tasks, and address book, where fields are filled out with random values or dummy text. Users with administrator privileges are further able to access the admin page and its subpages. On OkayCMS, users browse the articles available on the webshop and add or remove products from their shopping carts.

Figure 3.4 shows a sample transformation from TIM to TSM and further an exemplary execution of a parameterized system behavior script. The left side of the figure shows the state machine as well as the instantiation of a predefined number of profiles, each containing ranges of transition frequencies between the states, and their random allocation to users. The profiles also specify the browser used to access the websites. Moreover, the mail recipients are selected based on a randomly generated small-world network, i.e., most users communicate in small groups rather than randomly sending mails to every other user with the same probability [SHH$^+$06]. In addition, users regularly logout and

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

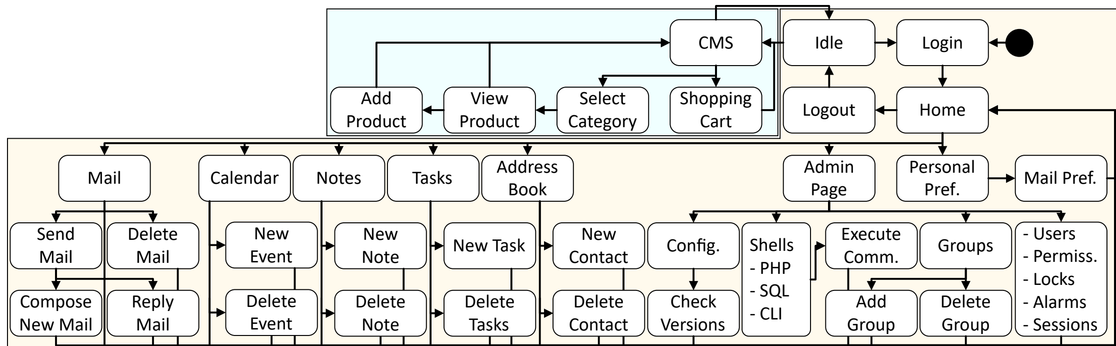


Figure 3.3: Simulated normal user behavior on the Horde Webmail (yellow) and OkayCMS (blue) modeled as a state machine.

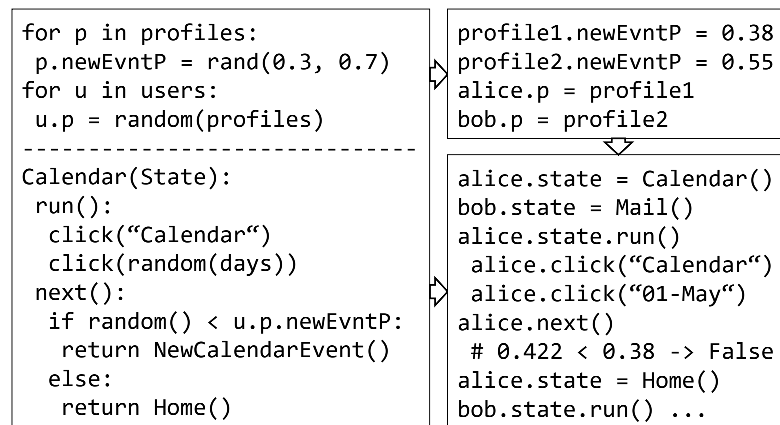


Figure 3.4: Randomized user profiles and state machine of system behavior TIM (left) transformed to TSM (top right) and testbed execution (bottom right).

go idle for random amounts of time, and stay inactive during night time to simulate daily routines.

The TSM generated by the transformation engine yields a configuration that is exemplarily displayed in the top right of Fig. 3.4. Note that probabilities for choices are normalized to ensure that they sum up to 1. All users exhibit different behavior, even though their actions are based on the same independent behavior model. The bottom right of the figure shows the execution log of users “alice” and “bob”. In this sample, user “alice” views a random day from the calendar, but returns to the home page rather than adding a new event, because a randomly selected value is below the threshold. This sample also shows multiple users using the system at the same time causing interleaving processes.

3.2.3 Attacker Behavior

We prepared two attacks to be executed on the testbed. The first one is a multi-step intrusion that involves several tools commonly used by adversaries and exploits two

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

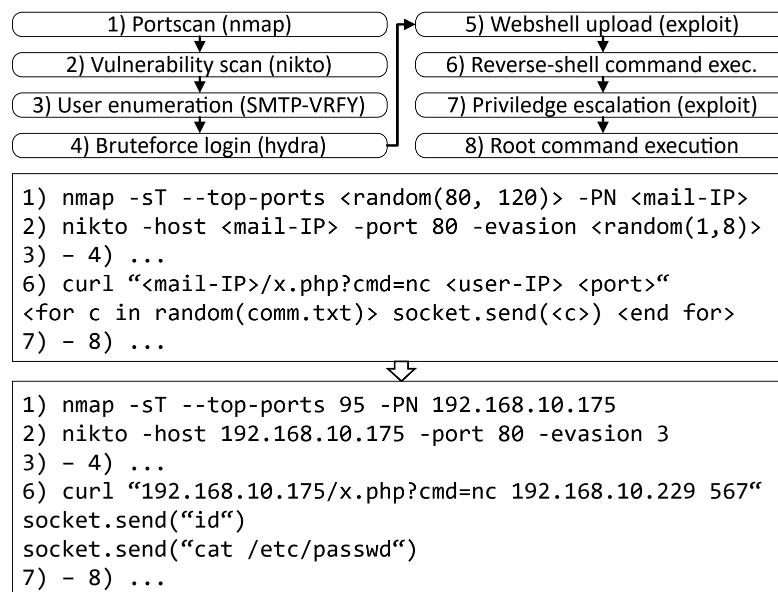


Figure 3.5: Multi-step attack on Horde Webmail (top) and its transformation from TIM (center) to TSM (bottom).

well-known vulnerabilities to gain root access on a mail server. The top of Fig. 3.5 shows an overview of the attack steps. The first two steps involve scans for open ports [Nma] and vulnerabilities [Nik]. Then, the attacker uses the smtp-user-enum tool [Smt] for discovering Horde Webmail accounts using a list of common names and the hydra tool [Hyd] to brute-force log into one of the accounts using a list of common passwords. The attack proceeds with an exploit in Horde Webmail that allows to upload a webshell (CVE-2019-9858) and enables remote command execution. We simulate the attacker examining the web server for further vulnerabilities by executing several commands, such as printing out system info. In our scenario, the intruder realizes that a vulnerable version of the Exim package is installed and thus uploads an exploit (CVE-2019-10149) to obtain root privileges through another reverse connection.

Figure 3.5 shows how we model this attack procedure as TIM and one possible transformation to TSM. As visible in the TIM, we use a sequence of predefined commands, but do not specify values that are only known after instantiating the testbed, such as the IP addresses of the web server ("mail-IP") and user host ("user-IP"), as well as parameters that are varied in each simulation, such as port numbers, evasion strategies, or commands executed after gaining remote access. This attack was purposefully designed as a multi-step attack with variable parameters to evaluate the ability of IDSs to disclose and extract individual attack steps and their connections, and recognize the learned patterns in different environments despite variations.

The second attack targets the web shop. A recently discovered flaw in OkayCMS allows an attacker to inject a malicious php-object via a crafted cookie (CVE-2019-16885).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In this scenario, the attacker uses the exploit to upload a webshell and is then again able to execute commands through the remote interface. Since no user credentials are required for authentication, this attack consists of only a single step and does not involve variations. Instead, it is designed to evaluate whether IDSs are able to detect and classify the injection of the php-object, since it only manifests itself in slightly different library calls that are difficult to detect.

Both attacks are carried out at a random point in time within a predefined period on randomly selected user host machines. Since the attacks are carried out independent of each other, they may be executed at the same time. During execution, the outcomes of the commands are automatically searched for keywords that indicate successful execution. We log this information together with the start and end times of each attack step, which is useful for labeling the recorded log data.

3.2.4 Ground Truth

Labeling data is essential for appropriately evaluating and comparing the detection capabilities of IDSs. However, generating labels is difficult for several reasons: (i) log data is generated in large volumes and manual labeling all lines is usually infeasible, (ii) single actions may manifest themselves in multiple log sources in different ways, (iii) processes are frequently interleaving and thus log lines corresponding to malicious actions are interrupted by normal log messages, (iv) execution of malicious commands may cause manifestations in logs at a much later time due to delays or dependencies on other events, and (v) it is non-trivial to assign labels to missing events, i.e., log messages suppressed by the attack.

We attempt to alleviate most of these problems by automatically labeling logs on two levels. First, we assign time-based labels to all collected logs. For this, we make use of the attack execution log mentioned in the previous section. We implemented a script that processes all logs, parses their time stamps, and labels them if their occurrence time lies within the time period of an attack stage. Under the assumption that attack consequences and manifestations are not delayed, it is then simple to check whether anomalies reported by IDSs lie within the expected attack time phases. Since exact times of malicious command executions are known, it is even possible to count correctly reported missing events as true positives.

While time-based labeling is simple and effective, it cannot differentiate between interleaved malicious and normal processes and does not correctly label delayed log manifestations that occur after the attack time frame. Therefore, our second labeling mechanism is based on lines that are known to occur when executing malicious commands. For this, we carry out the attack steps in an idle system, i.e., without simulating normal user behavior, and gather all generated logs. We observed that most attack steps either generate short event sequences of particular orders (e.g., webshell upload) or large amounts of repeating events (e.g., scans). We assign the logs to their corresponding attack steps and use the resulting dictionary for labeling new data. For the short ordered sequences, we pursue

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

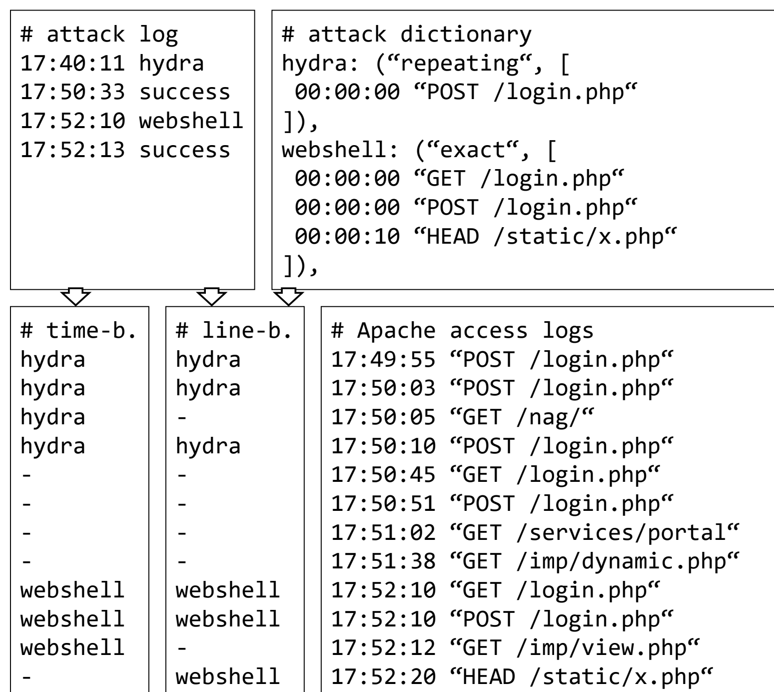


Figure 3.6: Example of our labeling procedure. Information on attack execution (top left) and expected attack logs (top right) are used to create labels (bottom left) using time-based and line-based techniques for log data (bottom right).

exact matching, i.e., we compute a similarity metric [WSL+17] based on a combination of string similarity and timing difference between the expected and observed logs and label the event sequence that achieves the highest similarity. For logs that occur in large unordered sequences, we first reduce the logs in the dictionary to a set of only few representative events, e.g., through similarity-based clustering [WSL+17]. Our algorithm then labels each newly observed log line that occurs within the expected time frame and achieves a sufficiently high similarity with one of the representative lines. These strategies enable correct labeling of logs that occur with a temporal offset or are interrupted by other events, but obviously suffer from misclassifications when malicious and normal lines are similar enough to be grouped together during clustering.

Figure 3.6 shows an example of our labeling procedure that involves two sample attack steps, the brute-force login tool “hydra” and the “webshell” upload. The top left of the figure shows start and end times of both attacks logged during attack script execution. The top right of the figure shows a dictionary that lists the log lines that are expected to occur in the Apache access log at attack execution. Note that the “hydra” logs are marked as “repeating”, i.e., they represent a large number of similar lines, while the “webshell” logs are marked as “exact”, i.e., they correspond to ordered individual lines. The bottom left of the figure displays the time-based and line-based labels for the Apache access logs in the bottom right. As visible in this example, the time-based labels are

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

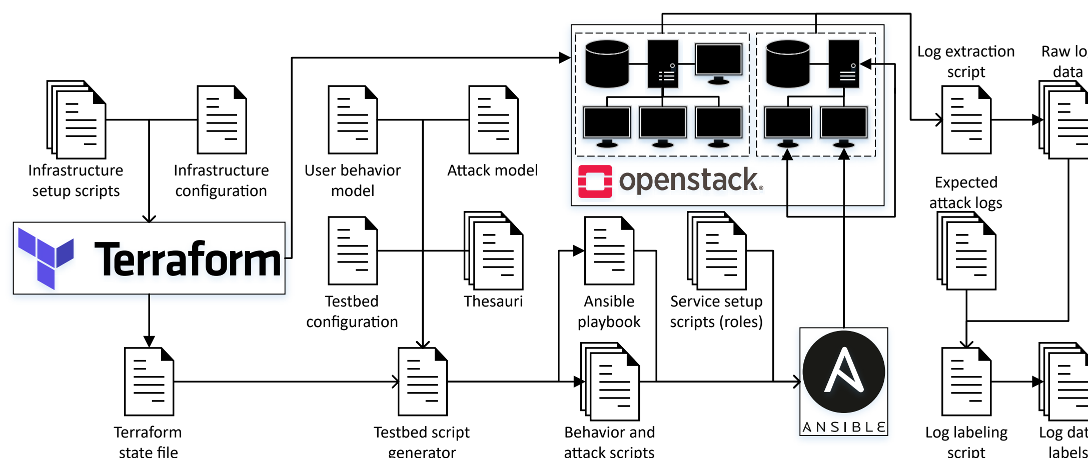


Figure 3.7: Technical implementation of the model-driven testbed and log data generation approach. Simple arrows indicate imports and filled arrows indicate generation of resources such as scripts, configuration files, or machines.

assigned to the lines solely by their occurrence timestamps. Due to the interleaving user actions, this means that lines generated by actions other than the attack (e.g., viewing Horde task list "nag"), but occurring in the same time frame, are also labeled accordingly. These lines remain correctly unlabeled by the line-based method. In particular, the "repeating" technique labels all lines within the "hydra" attack time frame that achieve a minimum string similarity to the message "POST /login.php". In this simplified example, these lines are identical and thus achieve a perfect similarity score. The "exact" technique matches the three expected lines of the "webshell" attack step within all lines occurring in the attack time frame to find and label their counterparts. Note that it is possible to specify the temporal offset through the timestamp in the attack dictionary, e.g., "HEAD /static/x.php" is expected to occur 10 seconds after the first two lines of the "webshell" attack step.

3.2.5 Implementation

We implemented the outlined concept for the automatic generation of testbeds using model-driven techniques. Figure 3.7 shows an overview of the typical workflow for testbed and log data generation. As visible in the figure, we use the infrastructure-as-a-service tool Terraform [Ter] to instantiate the testbed infrastructures as virtual machines on an Openstack [Opeb] cloud platform using our predefined setup scripts. Configurations at this point involve the total number of machines, operating systems, and computational resources, e.g., memory.

Building the machines with Terraform yields a so-called state file that contains deployment information, such as IP addresses. The testbed script generator implemented in Python that acts as the transformation engine of our model-driven proof-of-concept implementation imports the state file together with a configuration file, system behavior

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

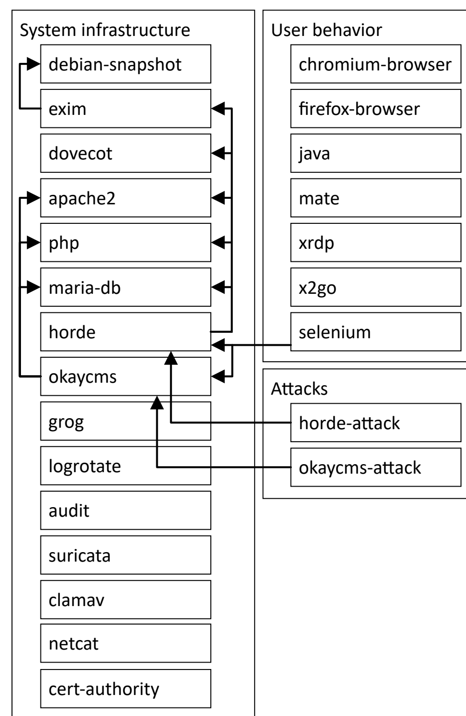


Figure 3.8: Overview of role dependencies. To install any role, it is necessary that all roles that the attached arrows point to are available and correctly installed.

and attack TIMs, and thesauri, i.e., word lists arranged by topics such as usernames, passwords, and host names. The configuration contains lower and upper limits for parameters that are randomly chosen when generating TSMs, i.e., files and executable scripts. Moreover, the transformation function generates a playbook that specifies the services to be installed, which are referred to as roles. Examples for such roles are PHP, Apache for web server setup, MariaDB for database setup, suricata IDS, or Internet browsers. Each role requires a setup script that states a list of tasks to be carried out. Thereby, it is possible to use variables in the playbook to specify random modifications of the setup process, e.g., install different versions, or replace them with alternative roles altogether. We then use the application-deployment tool Ansible [Ans] to distribute all generated files, set up services, and start the execution of user and attack scripts.

Note that roles have dependencies that have to be deployed before initiating the installation of the dependent role. Figure 3.8 shows an overview of all roles currently available and their dependencies. As visible in the figure, the roles for installing Horde Webmail and OkayCMS require several other roles for database setup, user managment, mailing services, etc. The user automation scripts as well as the attacks on the respective web services in turn require the availability of Horde Webmail and OkayCMS to access the web pages. In addition, some services are depending on specific versions, e.g., the vulnerable Exim version requires a specific debian snapshot.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The right side of Fig. 3.7 shows that once the simulation is complete, another script collects all log files from the virtual machines and stores them on disk. As outlined in the previous section, we automatically label the logs using attack execution information extracted together with the other logs as well as a predefined dictionary of expected log lines for each attack step. We store the lists of generated labels in separate files.

3.3 Validation

We devote this section to the validation of our approach and discovery of limitations. We first evaluate whether our approach adheres to the design principles defined in Sect. 3.1.1. We then analyze the collected log data and show the effects of automatically selected parameters on the system behavior. In addition, we discuss selected case examples to demonstrate the simplified process of iterative testbed development.

3.3.1 Fulfillment of Design Principles

We selected a simple web server to target a realistic and common use case. However, real web servers may be accessed by humans as well as bots with extremely high frequency and diverse behavior patterns. While our approach theoretically allows to add arbitrary numbers of user hosts, the total amount of machines is limited by the available computational resources and may thus not represent the heavy loads present in real networks. In addition, we did not use real user activity measurements to define behavior parameters, but argue that our model-driven approach makes it easy to adjust the TIM appropriately if such data is available. The prepared attacks are realistic, relevant, and make use of recently discovered exploits. Finally, all used log sources were either left in their standard configurations or were realistically adapted.

Our approach fulfills all three dimensions of the flexibility principle due to the incorporation of model-driven techniques. The number of testbeds and sizes of the networks only depend on the predefined amount of machines. Changing components or user and attack behavior is easy by modifying TIMs. For example, it is simple to extend the state machine that represents an independent model of the user behavior by adding new states for particular actions, while leaving everything else untouched. Since all the configuration files are reusable, it is possible to recreate the overall system behavior multiple times and thus reproduce the results. In addition, all technologies used in our scenario as well as the tools used to generate the testbed (Terraform [Ter], Openstack [Opeb], Ansible [Ans]) are open-source.

We made all produced log data available online in documented form. In addition, we provide labels for the logs created by time-based and line-based methods. The labels are on the level of attack steps and thus support the evaluation of IDSs. Finally, our generated data covers several days and thus contains several periods of repeating patterns, which allows anomaly detection tools to learn a baseline of normal behavior.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

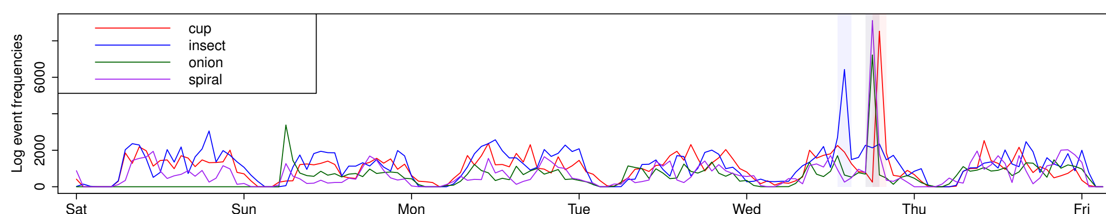


Figure 3.9: Event frequencies of Apache access logs. Scans executed as part of a multi-step attack manifest themselves as peaks (shaded intervals).

3.3.2 Manifestations of Testbed Variations

Depending on the types and characteristics of variations, different log files are affected in particular ways, e.g., by event appearances or changed parameters. In the following, we focus on event frequencies as a measure to compare testbeds. We analyze Apache access logs, because they keep record of page visits on Horde Webmail and OkayCMS, and thus allow to reconstruct user behavior, which is subject to variation.

Figure 3.9 shows user access frequencies on four web servers *cup*, *insect*, *onion*, and *spiral*, aggregated in time windows of one hour over six days. The plot depicts that users access the server more frequently during the day than at night, resulting in a daily cycle. The peaks (shaded intervals) are caused by the scans as part of the multi-step attacks. Note that additional detection techniques are required to disclose manifestations of the remaining attack steps. Since the amount of users per web server is selected randomly in order to increase variation, the average access rates differ among the web servers.

We further retrieve activity logs from each user. Since transition probabilities of the behavior state machines are specific to each user and remain constant over time, it is possible to relate observed behavior to users. For this, we compute the relative frequencies of accessed web pages for each user in time intervals of one day and use principal component analysis (PCA) to scale down the resulting high-dimensional data. Figure 3.10 shows a biplot [Bip] containing daily user behavior as scores (visualized as points) and the influence of visited pages on principal components as loadings (visualized as vectors). The ellipses represent the normal distributions of the daily user activities and show that each user follows a distinct pattern. The behavior spectrum includes admin users (*daryl*, *lacresha*, *lino*, and *sadye*) and shows overlaps between users following similar behavior profiles, e.g., *denis* and *long*.

We also extract page visits from Apache access logs collected at the web servers. Note that we do not attempt to trace individual accesses to specific users; instead, we analyze differences of the overall behavior observed at the web servers. Figure 3.11 is another biplot that shows groups of daily page visits on the web servers. Despite the aggregation of different user behavior patterns, the activities on the web server form distinct groups, for example, *onion* is located far away due to its high activity of admin users. The figures suggest that our approach achieved to generate testbeds with variations.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

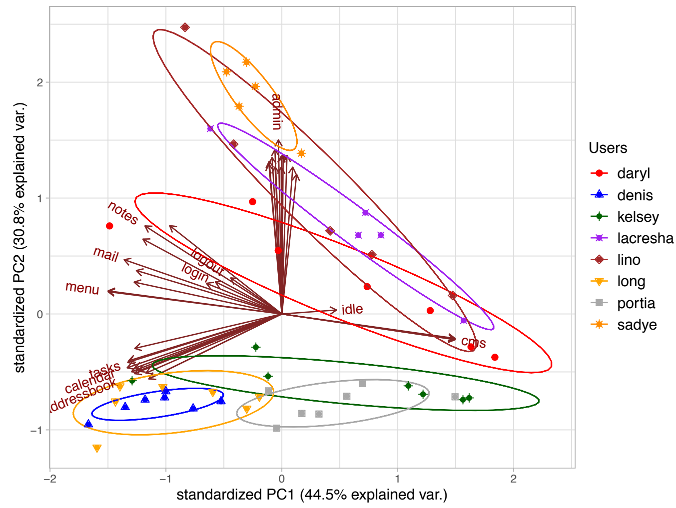


Figure 3.10: Biplot of user page visit frequencies aggregated in daily intervals.

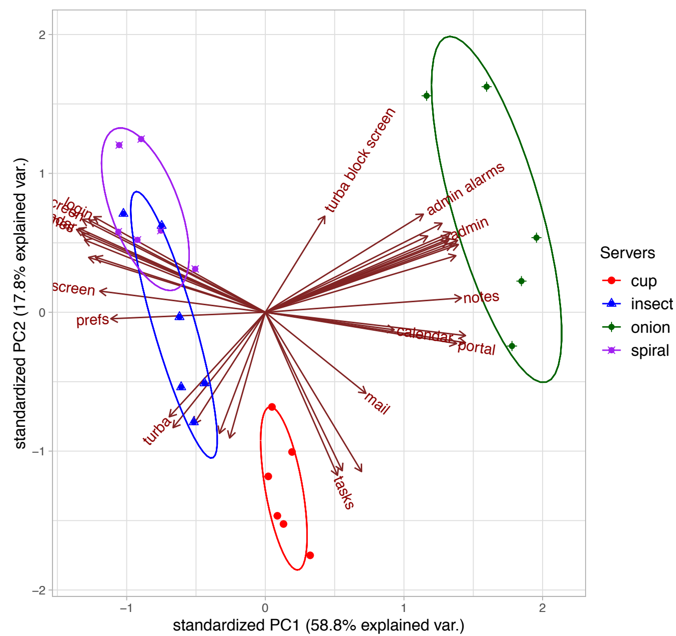


Figure 3.11: Biplot of Apache access logs collected from different testbeds.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

3.3.3 Case Examples of Testbed Extensions

We designed the presented approach to simplify iterative testbed development and support variations across generated testbeds. Our experiences during development of our proof-of-concept implementation (cf. Sect. 3.2) endorsed the achievement of this target. In particular, we proceeded by adding new TIMs as reusable modules and repeatedly built and destroyed testbeds to test new features. In the following, we discuss three case examples of such extensions in detail and measure the required manual work in lines-of-code that were adapted.

Tool

System administrators install different tools on web servers based on domain knowledge and personal preference. We selected the Clam AntiVirus software [Cla] as an exemplary tool to be installed on some testbeds. Since there are no dependencies to other modules, another infrastructure TIM with 10 lines of code is required to define a new role that contains two tasks that install the software and set up a cron job that regularly performs scans. Thereby, we leave the scheduled scan time as a variable. In the transformation engine that generates and populates the testbed setup scripts, we add 14 lines of code to specify the probability for installing Clam Antivirus, set the scan intervals, and add the resulting parameters to the Ansible playbook.

Browser

Since real users prefer different browsers for accessing web platforms, we planned to add Firefox [Firb] as an alternative to Chromium [Chr]. Similar to the antivirus tool, this implies creating a role with a single task consisting of 6 lines that specify the installation details. However, it is further necessary to change the existing TIM of the user behavior by adding a task that copies the required browser drivers for web automation (4 lines of code) and adapting the user behavior script to support the new browser (5 lines of code). Finally, a single line is edited in the transformation engine that randomly assigns one of the available browsers to each user profile.

Web Platform

At first, only Horde was implemented in the testbed. To increase diversity of the generated log data, we then decided to extend the simulation to also include OkayCMS [Oka]. For this, we first set up and configured an OkayCMS instance. Once this was accomplished, a role with only 23 lines of code was required to specify three tasks that copy the OkayCMS instance in the appropriate webroot and set up the database to make the web store accessible to the users. Around 120 lines of code were necessary to update the state machine in the user behavior TIM so that users are able to navigate four pages of the website and perform adequate actions. Finally, three lines of code in the transformation engine specify the transition probabilities between the states.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.4 Discussion

We propose to shift from traditional testbed setup to model-driven testbed design in order to overcome common issues, including high manual efforts and repeated work when adjusting or upgrading components. In the previous sections, we discuss several design aspects and show examples of TIMs and automatically generated TSMs. In the following, we will outline possible use-cases and review limitations that could provide ideas for future work.

3.4.1 Applications

There are several promising use-cases for our model-driven testbed generator. Foremost, our main intention is to automatically build testbeds for generating log datasets suitable for IDS evaluation without the need to start from scratch for every new use-case, but instead reuse existing components and develop testbeds iteratively. For example, starting from our proof-of-concept, it is possible to introduce and exploit new vulnerabilities by changing only the affected components and attacks, while leaving everything else untouched. Another idea is to model account hijacking by changing a user profile at some particular point in time, which could be the focus of detection tools based on user profiling.

Since testbeds are isolated from real networks and thus do not produce sensitive data that could raise privacy concerns, the generated log data is always suitable to be shared with others [UHH+21]. Moreover, it is simple to create multiple variants of the same testbed in parallel and generate several datasets that represent different environments. This improves the robustness of results from IDS evaluation and allows researchers to measure the variation of the detection capabilities of their IDSs.

Alternatively, it is possible to deploy IDSs directly in the generated testbeds by adding an appropriate setup script to the infrastructure TIM. In this case, the generation of log data is less relevant, and instead analysts are able to observe and measure the detection capabilities in real-time. This application scenario could be especially useful for experiments and live demonstrations, where attacks are injected manually.

Another relevant application case is malware analysis. Since it is easy to generate many testbeds with variabilities, inserting the code to deploy malware in the TIM allows to observe their behavior in different environments. This enables analysts to derive insights on the behavior of the malware without much effort spent on setting up the necessary machines. Then, the same attack can be deployed in testbeds with patched services to ensure that the intrusions fail in every case.

Finally, our provided datasets contain log data rather than network traffic and thus enable evaluation of host-based IDSs, a field where datasets are urgently needed [ČG18]. In addition, since one of our injected attacks involves the execution of several steps, the resulting dataset is a great benefit for the research community around multi-step attacks, where publicly available datasets are rare [NDP18]. Even more so, the variations of these

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

multi-step attacks across our generated datasets enable evaluation of algorithms that extract attack patterns independent of the environment and transform them into reusable cyber threat intelligence [LSW+19].

3.4.2 Limitations & Future Work

Despite the aforementioned benefits, we recognize some drawbacks of our method. First, it is necessary to point out that model-driven testbed design requires more effort than setting up a single static testbed, because all installation procedures have to be formalized and separated into fixed and variable parts that are subject to change, e.g., IP addresses have to be dynamically retrieved whenever they are necessary for a command. However, we argue that this increased initial effort pays off when testbeds are reused multiple times, especially when application scenarios are subject to change or multiple instances and variations of testbeds are required.

We further encountered that the ability to automatically upgrade all components to their newest versions in each rollout comes handy to ensure that the testbed is relevant to real world scenarios, but possibly causes problems when services are dependent on each other or rely on version-specific configurations. In such cases, there is no way around manually fixing the TIMs, because such requirements of future versions cannot be foreseen. For critical components, it is possible to always install a fixed version, despite the downside that the service will eventually be outdated.

It is also important to note that generating data in our generated testbed requires the users to run in real-time. The reason for this is that it is infeasible to speed up the actions carried out by users, e.g., decreasing the sleep time between commands in the user behavior or attack scripts, since also the timestamps have to be adopted accordingly, i.e., the generated log data needs to be modified in hindsight. In addition, properties of the infrastructure, e.g., latencies and loading times, may have unrealistic influence on the log data when timestamps are changed, and eventually limit the possibility to increase the speed of the publication. Thus, it is not simply possible to simulate long timespans in a short amount of time.

Another limitation of our approach is that our line-based method for automatically labeling log messages corresponding to malicious activity is not guaranteed to always yield correct results and should thus only be seen as a complementary approach to the time-based method that provides additional confidence to the labels. The reason for this is that this method is based on string similarity, and as such is unable to differentiate between messages that are not sufficiently distinct, which leads to incorrect labeling. In addition, selecting the similarity threshold is non-trivial, since it depends on the overall structures of all possible log events. At the moment, gathering the expected logs for each attack step involves manual work, in particular, executing each attack step separately to populate the attack dictionary. Introducing new attack steps or changes of the logging infrastructure require to repeat this process. In Chap. 4, we therefore

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

present a methodology to ease this task. Nevertheless, attack steps that involve random or otherwise variable manifestations will remain difficult to label correctly.

Regarding the log dataset produced in our proof-of-concept, we see a number of extensions that could improve future simulations. First of all, the authenticity of the user behavior can be improved by deriving parameters from real system usage, which was omitted due to lack of such real data. In addition, randomness is usually based on uniform distributions, however, actual user behavior could be better represented by other distributions, such as the normal distribution.

Finally, it would be interesting to develop a formal modeling language for generating testbeds. Thereby, the transformation engine would work as a function that selects properties of infrastructure components, user behavior, and attacks, from the predefined ranges of allowed values. This would help to define a metric that makes testbeds comparable by measuring their similarity through their common properties. Aggregating such a testbed similarity metric over all generated testbeds, it would be possible to provide the analyst with a feedback on the diversity of the testbeds, i.e., a measure on the coverage of possible combinations of model parameters. Ultimately, the resulting aggregated metric could be used to determine whether an appropriate amount of testbeds have been generated to represent most possible testbed configurations, or to calculate an estimation for the number of testbeds required.

3.5 Summary

In this chapter, we proposed a methodology for creating testbeds for log data generation using techniques from model-driven engineering. For this, we designed abstract models for the testbed infrastructure, the simulated system behavior, and the injected attacks, and used a transformation engine to automatically translate these testbed-independent models (TIM) into testbed-specific scripts and configuration files that allow deployment. This increases the required initial effort, but largely reduces the amount of work required to maintain and modify testbeds for different application scenarios. Due to the fact that testbed-independent models (TIM) only define parameters as discrete lists or ranges of allowed values, we were able to generate arbitrary numbers of testbeds with variations.

While the deployment and configuration of the testbed as well as the collection of log data is fully automatic, labeling of log events that are related to attacks requires human effort for gathering and maintaining the attack dictionary. In addition, the proposed similarity-based approach is tricky to configure and may assign incorrect labels as pointed out in Sect. 3.4.2. The following chapter will therefore extend the proposed approach with an automatic labeling procedure that leverages rules rather than similarity-based matching.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 4

Log Data Generation & Labeling

The previous chapter introduced a method for the automatic creation of simulation testbeds. These testbeds are the basis for the event labeling methodology presented in this chapter. In addition, the log datasets analyzed in this chapter are generated and labeled on such a testbed. Major parts of this chapter have been published in [LSF+] and [LFS+22].

As outlined in the previous chapter, testbeds have several benefits for log data generation in comparison to real infrastructures, including the abilities to arbitrarily adjust configurations, launch attacks without worrying about system damage, and make generated datasets publicly available since they are free of sensitive data. Several log datasets have therefore been generated in such controlled environments in the past. However, one of the main problems that is rarely addressed in the methodologies for generating these datasets is reliable labeling of malicious events, even though it is crucial for any evaluations with respect to attack detection.

In general, the typical process of labeling log data as realized by existing works makes use of similar techniques as signature-based IDSs: Log events are scanned for particular keywords, e.g., IP addresses of attacker hosts, and labels are assigned to matching logs to categorize them as malicious [AAAH+18]. This is intuitively reasonable for simple scenarios, for example, where all activities originating from a dedicated attacker machine are known to be malicious. In testbeds where analysts have full control and information about the simulated attackers, labeling is trivial for such cases [RWG+17]. However, more complex and realistic testbeds involve attack manifestations that are interleaved with traces of normal system behavior, which makes it difficult to discern these two classes. In particular, labeling by simple IP-based matching is impossible when normal and malicious activities are executed simultaneously on the same host, which necessarily occurs when attackers manage to compromise and misuse actively used systems. Even worse, system logs that are necessary to evaluate HIDSs rarely contain network information and do not even need to involve any expressive descriptors for keyword matching; in fact,

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek, Your knowledge hub, WIEN

manifestations of attacks are possibly identical to normal events and only their combined occurrences in a specific execution context allow to determine their root cause. In addition, system logs are usually unstructured and generated in heterogeneous formats so that labeling rules cannot simply be applied on all log files [ZHL+19]. Unfortunately, common ways of data labeling based on keyword matching are thus unable to adequately label logs for HIDS evaluation.

Data generation methodologies that leverage model-driven testbeds as described in Chap. 3 suffer from the same problem [GFDVC10, LSW+21]. In fact, the dynamic assignment of variables such as IP addresses imply that labeling rules based on hard-coded values cannot simply be reused across testbeds, since these values are purposefully subject to change. Therefore, it is necessary to repeatedly adapt labeling rules, creating a bottleneck for model-based testbed generation.

This chapter aims to resolve aforementioned problems by integrating labeling rule design into the model-driven testbed generation process. For this, our framework utilizes abstract labeling rule templates that are completed with automatically extracted testbed parameters. In addition, we propose four rule types that allow to label system log datasets for HIDS evaluation. We also publish a collection of log datasets generated with the presented approach as well as all code that is necessary to run our testbed and simulations within it so that other researchers are able to replay or augment the simulation runs. Our datasets are therefore maintainable and allow for continuous improvements such as enlargements of the labeling range as well as additions of datasets from new testbeds. We summarize our contributions as follows:

- A framework for model-driven labeling of system log data,
- a publicly available labeled collection of log datasets for evaluation of IDSs,
- an analysis and comparison of these datasets with respect to real user logs,
- an open-source implementation to launch testbeds for dataset generation, and
- an open-source library and models to simulate normal user behavior and attacker activities.

The remainder of this chapter is structured as follows. Section 4.1 outlines an integrated concept for model-driven testbed generation and log data labeling. In Sect. 4.2 we outline our methodology for generating log datasets and explain our modeled scenario. We analyze the generated datasets in Sect. 4.3 and discuss the results in Sect. 4.4. Finally, Sect. 4.5 summarizes the chapter.

4.1 Methodology

Chapter 3 presented a model-driven procedure for testbed generation that utilizes separate models for infrastructure, normal system behavior, and attack executions. This procedure

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

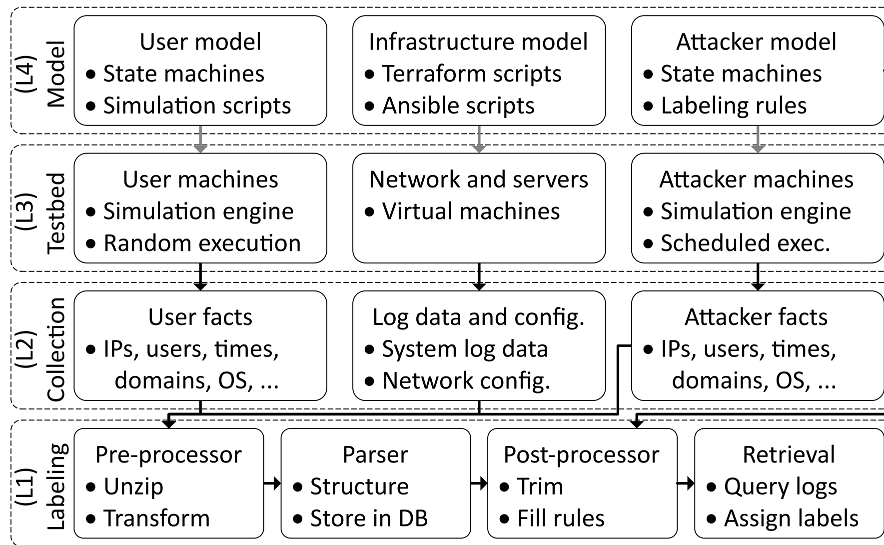


Figure 4.1: Labeling concept for model-driven testbeds.

is extended and reflected in Fig. 4.1, where layer (L4) represents the TIMs, i.e., models consisting of templates for deployment scripts, configurations, and state machines. Layer (L3) holds TSMs, i.e., specific instances of testbeds. Note that these two layers represent a simplified form of the procedure depicted in Fig. 3.1. As such, the TSMs allow to roll out the testbed and run the simulation. We refer to the previous chapter for more information on these steps.

Once the simulation is completed, the collection layer (L2) handles the extraction of relevant information from the components. Foremost, this concerns the log data to be labeled, which is collected from machines that are typically monitored by IDSs, e.g., web servers and firewalls. In addition, we collect logs relevant for labeling from all other machines in the testbed. Thereby, logs from the attacker machine that outline the timeline of attack executions are likely the most important source of information; however, also logs from hosts running simulations of normal user behavior can be used to verify attack labels or assign labels to benign activities. Moreover, we gather system information that we refer to as *facts* from all components. Facts are the main source for filling out labeling rule templates and include artifacts such as IP addresses, domain names, user names, OS versions, etc [GSL+20]. Finally, we also extract configuration files of installed services and logging frameworks. All gathered data is then transmitted to a central storage system where labeling takes place.

The labeling layer (L1) generates ground truth labels for datasets in a four-step procedure. First, a pre-processor prepares the logs for further analysis. In particular, this includes unzipping log files that are compressed during rotation and converting logs stored as binary or other formats into raw text files. Second, a parser transforms the system log data, which is usually only available in unstructured form, into a semi-structured format

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

so that all tokens can be referenced individually. These parsed log events are then loaded into a database that supports storing and searching such semi-structured data. Third, a post-processor trims the logs to fit the desired simulation time interval and prepares all rules by inserting facts extracted by the collection layer into rule templates designed as part of the attacker model. Fourth, the retrieval step iterates over all rendered labeling rules and executes queries on the database storing the parsed log data so that all matching events are assigned one or more labels corresponding to the respective rule. Sect. 4.2.4 will provide more details on these labeling rule templates, in particular, an overview of different query types.

4.2 Scenario

The previous section outlined a general overview of the methodology for the generation of our dataset. In this section, we first describe our targeted use-case and explain specific design decisions regarding variations in the dataset before presenting the approach for data labeling in detail.

4.2.1 Use-case

The purpose of our collection of log datasets is to enable evaluation of IDSs in the context of a widespread application scenario that is frequently subject of cyber attacks. Specifically small- or medium-sized organizations are a frequent target of cyber attacks, often due to the fact that they do not have the required resources for extensive protection [Sym19]. We therefore design our testbed to resemble a small enterprise network that follows well-known security guidelines, such as segmentation of networks into zones [ISO10]. This is one of the aspects that we improved upon in comparison to the testbed infrastructure described in Sect. 3.2.1.

Figure 4.2 displays an overview of the network realized by our testbed. The network comprises three zones: (i) the intranet that contains a number of Linux hosts [Ubu] for each employee as well as an intranet server running WordPress [Wor] and Samba file share [Sam], (ii) the demilitarized zone (DMZ) that contains servers for VPN [Opec], proxy, Horde Groupware [Hor], and cloud share [Own], and (iii) the Internet with global DNS [Mar, Dnsa], hosts for remote employees that connect to the intranet via VPN, external employees that use external mail servers, and an attacker host. The zones are connected via a firewall [Sho] that also acts as an internal DNS server for all domains owned by the organization. All employed technologies are publicly available and commonly used in real networks [MFCMC+18].

As outlined in Chap. 3, TIMs result in different TSMs due to the fact that several parameters are set dynamically during instantiation of the testbeds. With respect to the system environment, this mainly concerns the network size and allocation of IP addresses. In particular, we generate between 3 and 9 hosts for internal, remote, and external employees respectively, meaning that the final testbed may consist of at least

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

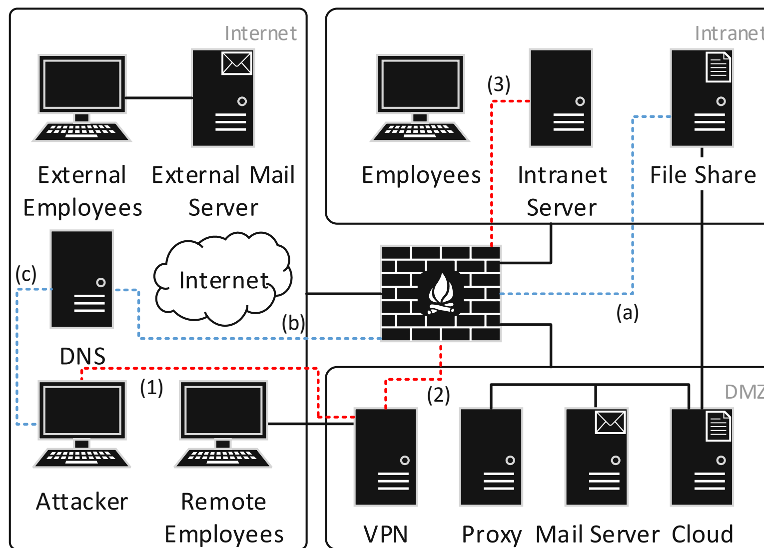


Figure 4.2: Overview of the testbed network. Steps (1)-(3) mark the attacker's path to compromise the intranet server and steps (a)-(c) represent connections related to the data exfiltration attack vector.

Table 4.1: Variations of the system environment

Parameter	**Range**
Number of user hosts	9-27
Number of mail servers	2-4
Network zone classes	[a, b, c]
Host IPs	Random IP within respective zones
Network and zone names	Random names

9 and at most 27 user simulations running in parallel. Similarly, we generate between 2 and 4 external mail servers. We also assign each network zone a random class and randomly choose IP addresses from these zones for each host. Finally, we also configure the domain names of all network zones as random names using the Faker library [Fak]. Table 4.1 provides a summary of all variations of the technical infrastructure.

4.2.2 User simulation

Real networks in small- or medium-sized organizations are actively used by humans that carry out their daily routines in their workplace. The simulation of normal behavior is therefore an essential aspect of synthetic dataset generation for IDS evaluation. Simulated normal system behavior that is not sufficiently complex may result in non-representative datasets that yield too low false positive rates during IDS evaluation, as human interactions with machines are often erratic and possibly lead to unexpected system states that may

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

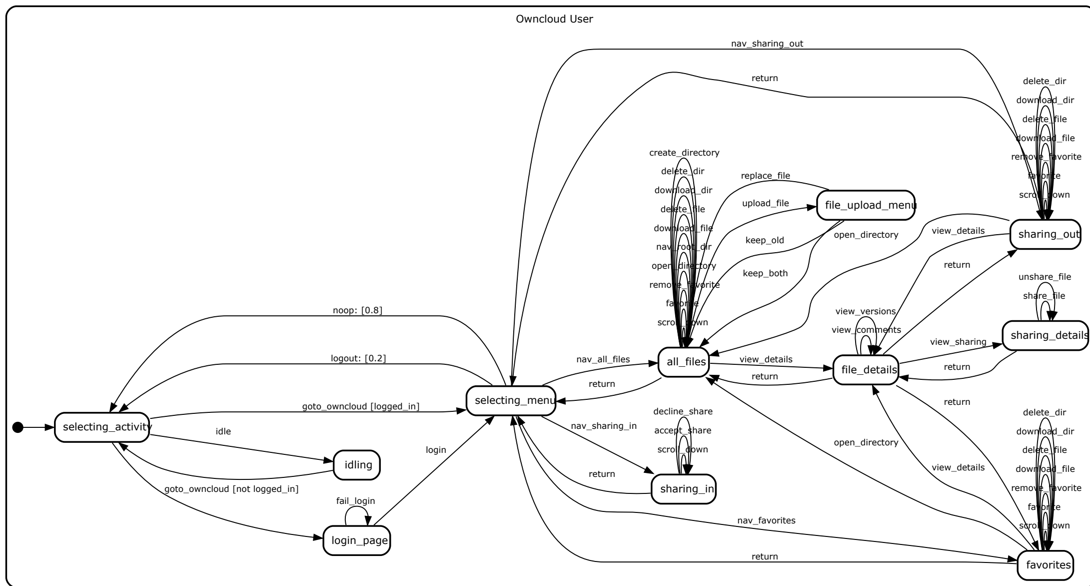


Figure 4.3: User state machine for simulating normal behavior on the cloud share platform.

be incorrectly detected as malicious. We therefore decided to create state machines for all services in our testbed that are normally accessed by real users. For this purpose, we make use of web automation software [Sel] that allows to use scripts to navigate on websites and click on specific links.

Figure 4.3 visualizes the state machine for a user accessing the cloud share platform. Note that states describe the current view of the users and that activities such as clicking buttons are carried out when traversing from one state to another. As visible in the figure, the user first logs into the OwnCloud platform (possibly with incorrect credentials, in which case login is retried) and then enters pages showing either all their files, files marked as favorites, files shared with other users, or files other users shared with them. Depending on their selection, the users are then able to view files, upload and share new files, change or remove existing shares, accept or decline invitations to share files, and manage their favorites. Furthermore, there is the possibility that a user leaves the cloud sharing application and switches to another website, or enters the idle state in which case no action is carried out for a certain amount of time. We argue that the total number of possible transitions and interweaving of states visible in Fig. 4.3 is sufficiently complex to represent real user interaction. Section 4.3.2 will compare log data generated by simulated and real users to verify that their access frequencies and usage distributions are indeed similar.

We do not provide figures for all state machines for brevity, but briefly discuss their main features. (i) The web mail state machine (cf. Sect. 3.2.2) allows users to view, compose, and respond to mails from other users, attach files to mails, change their preferences,

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 4.2: Variations of simulated user behavior

Parameter	Range
User name	Random name
Password	Random string
Wordpress role	[editor, admin, none]
SSH admin	[yes, no]
Samba role	[employee, mgmt., acc., admin, none]
OwnCloud role	[employee, mgmt., acc., admin, none]
Working hours	(5:00-9:00) - (17:00-22:00)
User mail provider	Random selection from all mail servers
User mail contacts	Random selection from all users
State transition probabilities	0.0-1.0
Web browser	[firefox, chromium]
Idle times	Tiny: 0.4-2.5 seconds Small: 3-60 seconds Medium: 40-360 seconds Large: 400-3600 seconds

and manage their calendar entries, contacts, notes, and tasks. In addition, privileged users may access the administrator panel to view and change settings of the platform. (ii) The WordPress state machine allows users to read existing posts on the WordPress instance, publish new posts, comment on existing posts, and view available media. (iii) The Internet state machine allows users to browse the Internet by randomly clicking on links on one of the websites from a predefined list. (iv) The SSH state machine allows users to connect to a host in the network via SSH to execute some of a predefined list of commands. All state machines are connected with each other, i.e., users are able to change between state machines, to further increase the complexity of the simulation.

Whether a user accesses specific states within the state machines or not depends on their roles, which are subject to variation. In particular, we define an SSH administrator role and furthermore differentiate between editor and administrator roles on the WordPress page and employee, management, accounting, administrator roles on Samba and OwnCloud pages. When no role is assigned to a user, the respective state machine is not entered at all. The names of all users are randomly generated from databases and their passwords are random strings. We also vary their working hours, assign their preferred web browser, generate their mail addresses from one of the external mail servers, and select random samples for their usual contacts and available files. To ensure that all files involved in the simulation appear realistic and do not only involve completely randomized contents, we make use of a collection of predefined dummy files with non-sensitive contents. Table 4.2 provides an overview of the varied parameters and their parameter spaces. Note that we use idle times to temporarily pause the state machines not only in idle states that are specifically created for this task, but also when entering or leaving certain states.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

This accomplishes to simulate delays between single clicks (tiny), pauses for reading and reacting to website contents (small and medium), or longer breaks of inactivity (large). The table leaves out several minor parameters, such as limits for maximum daily accesses or factors that make repeated executions of same activities more unlikely, for which we refer to our open-source implementation.

4.2.3 Attack scenario

While simulation of normal user activity is necessary to ensure authenticity of the underlying conditions, injected attacks are required to provide the artifacts to be detected or classified by IDSs. Accordingly, it is essential to design relevant attack cases that fit the overall use-case and are suitable to generate desired consequences in the dataset. For our use-case, we decided to model a multi-step attack that involves several stages of a typical cyber kill chain [Fira] and makes use of common penetration testing tools [Kal]. The selected attacks are an evolution of the attack chain described in Sect. 3.2.3, as several attack steps are replicated, but adapted to fit the updated network structure. Figure 4.2 shows the connections and affected hosts of this attack scenario. In particular, steps (1)-(3) show how the attacker first accesses the intranet over VPN to gather information and eventually takeover the intranet server, and steps (a)-(c) indicate how data is extracted from the file share in the intranet zone over a public DNS server to the attacker. In the following, we explain all attack steps in detail.

As part of our attack scenario, we assume that the attacker illegitimately obtained VPN credentials that allow them to access the network. In real-world attack cases, obtaining such credentials could be achieved through phishing attacks or by compromising a personal computer of an employee. Note that we do not simulate this part of the multi-step attack, since it occurs outside of the enterprise's network and thus does not leave any traces in the logs.

Once the attack execution starts, the attacker makes use of the VPN credentials to remotely establish a connection to the network over the VPN server. The first step of the attack chain then consists of several scans of the network. In particular, the attacker employs the well-known tool Nmap [Nma] to carry out DNS and port scans in the DMZ network where the VPN server is located. This allows the attacker to discover the CIDR of the intranet network and thus extend their scans to the hosts located in the intranet zone. Eventually, a web service scan shows a WordPress instance running on the intranet server, which leads to the attacker selecting this server as a possible target for intrusion. The attacker thus launches a brute force directory scan using the tool dirb [Dir] in order to find potentially interesting files. Since this scan shows up no results that allow the attacker to progress any further, they carry out a WordPress security scan using the tool WPScan [WPS] in order to discover vulnerable versions or misconfigurations of plugins or themes installed on the server. Other than the directory scan, this security scan shows that a vulnerable version of the plugin wpDiscuz is present on the server. At this point, the attacker stops scanning and instead focuses on exploiting the vulnerability, which marks the end of the reconnaissance phase.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

By exploiting the vulnerable plugin, the attacker is able to perform unrestricted file uploads (CVE-2020-24186). This allows the attacker to upload a PHP webshell as a backdoor that in turn allows them to execute arbitrary commands with the privileges of the *www-data* user of the web server. The attacker proceeds to execute several commands to gather information about the host, e.g., reading out processes, command histories, OS information, connections, or file names. Eventually the attacker finds the password to the user database in the WordPress configuration file and is thus able to access all user names and their hashed passwords.

The attacker then attempts to crack one of the hashed passwords using a list of common passwords. For this, our attacker state machine branches into two paths. In one path, we assume that the attacker transfers the password hashes to their own system and manages to crack one of the passwords there. Since this activity takes place outside of the monitored network, no logs are created and thus detection is not possible. Accordingly, we simulate this case by simply pausing the state machine for a specific amount of time. The other path simulates that cracking takes place at the compromised server. For this, the attacker installs the tool John the Ripper [Joh] and uses a common password list for cracking. Due to the fact that the purpose of our datasets is to provide detectable traces of anomalous behavior, we opt for the latter case when running our simulations. Note that as part of our attack scenario, we assume that the password of at least one system user is always present in the password list and thus successfully cracked after a certain amount of time. Subsequently after obtaining the password, the attacker uploads a fully interactive reverse shell and misuses the compromised user account to escalate their privileges to root level. The attacker then executes several commands of which some require root privileges, such as reading out the shadow file.

As a final step of the attack kill chain, the attacker runs the DNSteal [DNSb] tool that exfiltrates sensitive data from the file share located in the intranet zone. Thereby, the tool starts a process that converts files from certain directories into base64 to conform to the requirements of DNS queries, splits them into chunks, and sends them as DNS requests through the firewall to a specific attacker-controlled domain in the global DNS. Eventually the data is transferred from the malicious domain to the attacker's host, where it is decoded and stored. While we could have modeled the attack chain in a way so that the attacker would set up this exfiltration tool once they gained system privileges, we decided to separate this step from the remaining attack vectors and instead start the exfiltration tool already at the beginning of the simulation. The reason for this is that we decided to design the exfiltration attack as a challenge for anomaly-based IDSs that usually rely on an training phase that is free of attacks. By running the tool from the beginning of the simulation, we purposefully poison the training phase so that the malicious DNS communication is learned as part of the normal system behavior. However, the attack may still be detected by anomaly-based IDSs, since the exfiltration stops after a few days when all files are extracted. This is especially challenging, since it is usually more difficult for an IDS to recognize that a service suddenly stopped compared to the detection of a newly started service.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Table 4.3 summarizes the attack scenario. The first column maps each of the attack steps stated in the second column to phases of the cyber kill chain [Fira]. As stated before, the *Data Exfiltration* step does not chronologically follow the other attack steps. The third column lists related tactics and techniques from the well-known MITRE ATT&CK matrix version 10 [MIT] for each attack step. The matrix classifies and describes a wide range of common attack techniques and also provides information on detection. As visible in the table, our multi-step attack involves a diverse set of attack techniques that are part of several tactics. Finally, the last column states the most relevant log files that contain attack traces for each attack step. Since many different log files are affected, it is necessary to configure IDSs to monitor several hosts of the network in order to obtain a full picture of the multi-step intrusion.

Similar to the infrastructure and user behavior, we vary the attack parameters as part of the transformation from TIM to TSM. Table 4.4 provides an overview of the main variations used to generate the dataset. Note that while the time of day at which attack execution is initiated is varied, we manually set the day for each simulation run in advance. The reason for this is to avoid that the attack is launched too early and thus the dataset does not provide a sufficiently long training phase of at least 3 days. To select and implement variations of parameters of utilized attack tools, we looked up allowed values and ranges for each parameter in the respective documentations. Since tools such as WPScan [WPS] and DNSteal [DNSb] have multiple parameters that support ranges of allowed values, many possible combinations of values exist and thus the attack traces resulting in the logs are highly different. To realize random command executions, we assembled a list of common commands and randomly sampled them. We also injected the user password to be cracked in specific positions of the password file used by John the Ripper [Joh] so that the duration to complete cracking varies in each run.

These variations of the attack scenario are one of the reasons why labeling of specific events is difficult when log data is generated from model-driven testbeds. In particular, the labeling rules need to be flexible enough to cover different consequences of injected attacks. In addition, variations of the system infrastructure contribute to the resulting complexity of attack manifestations in log data. The following section therefore presents labeling rules that align with the model-driven procedure of our approach and enable labeling of different types of attack consequences.

4.2.4 Labeling

The previous section described attack cases for our testbed scenario. In this section, we define rules for labeling the log events generated as part of these attacks. For this purpose, we first propose four rule types that are suitable to be applied in model-driven settings. We provide selected examples for these rules that enable label assignment for the injected attacks. We then outline a strategy to select appropriate rule types and discuss our implementation of the approach.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 4.3: Overview of the attack scenario

Kill chain phases	**Attack steps**	**Tools**	**MITRE ATT&CK Tactics and Techniques**	**Data sources**
Reconnaissance	Traceroute Network scan DNS scan Service Scan	Nmap [Nma]	Reconnaissance - Active Scanning - Gather Victim Network Information	DNS logs Network traffic
Reconnaissance	WordPress scan Directory scan	WPScan [WPS] Dirb [Dir]	Reconnaissance - Active Scanning - Gather Victim Host Information	Access logs Error logs Network traffic
Initial Intrusion Establish a Backdoor	Webshell upload Webshell command execution	Shell	Execution - Exploitation for Client Execution Persistence - Server Software Component Discovery	Access logs
Obtain User Credentials	Wordpress database dump	Shell	Credential Access - OS Credential Dumping	Access logs
Obtain User Credentials Install Various Utilities	Password cracking	John the Ripper [Joh]	Credential Access - Brute Force: Password Cracking	Monitoring logs
Privilege Escalation	Login as system user	Shell	Privilege Escalation - Valid Accounts	Auth logs Audit logs
Lateral Movement	Reverse shell setup Root command execution	Shell	Execution - Command and Scripting Interpreter	Auth logs Audit logs
Data Exfiltration	Exfiltration over DNS	DNSteal [DNSb]	Exfiltration - Exfiltration Over Alternative Protocol	DNS logs Audit logs

Table 4.4: Variations of the attack scenario

Attack	Parameter	Range
General	Start times	00:00 - 24:00
	Attacker name	Random name
Network scans	Ports	100-2000 top ports
	Hosts	Random selection of servers
Wordpress scan	Scan mode	[passive, mixed]
	Enumeration	Random selection of plugins, themes, configs., database exports, users, and media
Directory scan	Recursive	[yes, no]
	Case-sensitive	[yes, no]
Webshell	Shell name	Random string
	Commands	Random commands
Password hash cracking	Mode	[online, offline]
	Duration	30-90 minutes
Reverse shell	Port	1100-65000
	Commands	Random commands
Exfiltration	DNS domain	Random string
	Forced IP	[yes, no]
	Compression	[yes, no]
	Verbosity	[yes, no]
	Block size	32-63
	Sub domains	integer of (200 / block size)

Labeling Rule Templates

As outlined in the beginning of this chapter, common labeling strategies are usually centered around searching for specific keywords, e.g., the IP address of an attacker machine, and marking all matching log events as malicious. Unfortunately, this is not possible in model-driven testbed generation approaches, because these keywords are not available at the time of designing the TIMs. Furthermore, system log data is not always discernible by such keywords and only combined and contextual occurrences of events allow correct label assignment. To overcome these issues, we propose labeling rule templates that are designed on the same level of abstraction as TIMs and are thus independent of artifacts specific to TSMs. In addition, we propose four types of rules to enable the assignment of labels to events that could not be labeled with common labeling strategies: *query*, *sequence*, *sub query*, and *parent query* rules. In the following, we describe each rule type in detail and provide examples.

Query Rule. Query rules are the most basic type of labeling rule template. Their purpose is to match collected facts with specific parts of log events and assign labels to all retrieved logs. Accordingly, this type of rule is only applicable when all relevant logs

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
- type: elasticsearch.query
  id: dnsteal.domain.match
  labels: [dnsteal]
  index: [dnsmasq-inet-firewall]
  query:
    bool:
      should:
        - regexp:
            dns.answers.name: '.*\.{{ attacker.dnsteal.domain | replace('.',
                '\.') }}'
        - regexp:
            dns.question.name: '.*\.{{ attacker.dnsteal.domain | replace('.',
                 '\.') }}'
```

Figure 4.4: Query labeling rule that matches domain names.

are known to match the respective fact in the selected field, and no logs generated by benign behavior yield matches in that field. For example, in the simple case where all activities associated with a malicious domain should be labeled as part of an attack, it is possible to label a log file that monitors DNS connections by matching the domain name occurring in the logs with the attacker's domain name that was previously extracted as a fact [AGR+19].

Figure 4.4 shows an exemplary query rule for this case. The rule specifies that all logs in the *dnsmasq-inet-firewall* index matching the malicious domain referenced by variable *attacker.dnsteal.domain* in the fields *dns.answers.name* or *dns.question.name* are assigned the labels *dnsteal*. Note that variables such as *dns.answers.name* are resolved by the respective field of the parsed logs in the database where the query is executed, while *attacker.dnsteal.domain* is a templated variable as indicated by the curly braces and thus replaced by the respective fact when the rule is rendered from the template. Thereby, the domain name is a random string extracted from the TSM for attacker behavior. As visible in the rule, queries may be connected with boolean operators, e.g., the keyword *should* represents a logical *OR* operation. In addition, it is possible to apply functions on the terms, e.g., we use a replace function in the sample rule to escape dots and enable matching with regular expressions.

Sequence Rule. Some attack artifacts in log data cannot be labeled with query rules, but require a more advanced strategy. In particular, this concerns logs that can only be identified as part of the attack by their collective occurrence, while each of the events individually is indiscernible from logs related to benign behavior. We therefore propose sequence rules to model such cases.

Figure 4.5 shows a sample sequence rule that labels two consecutively generated log events from two different sources, packet capture (PCAP) stored in index *pcap-attacker_0* and Apache access logs stored in index *apache_access-intranet_server*. We use the *by* parameter to obtain groups of logs with the same value in the *url.full* field and set the maximum time span in which these logs have to occur through parameter *max_span* to

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
- type: elasticsearch.sequence
  id: attacker.foothold.apache.access
  labels: [attacker_http, foothold]
  index: [pcap-attacker_0, apache_access-intranet_server]
  by: url.full
  max_span: 3m
  filter:
    - range:
        "@timestamp":
          gte: "{{ ( foothold.start | as_datetime) }}"
          lte: "{{ ( foothold.stop | as_datetime) + timedelta(seconds=1) }}"
  sequences:
    - '[ apache where event.action == "access" and source.address == "{{
        attacker.vpn_ipv4_address }}" ]'
    - '[ http where source.ip == "{{ servers.intranet_server.
        default_ipv4_address }}" and layers.http.http_http_response == true
        ]'
```

Figure 4.5: Labeling rule of sequence type with filtering.

3 minutes. The sequence itself is specified through a list of queries, where each query matches log event fields to facts or predefined values, e.g., *source.address* of Apache access logs matches the attacker's VPN IP extracted from the infrastructure TSM. While this sample demonstrates labeling of events occurring across different files, we point out that this rule type is also highly useful to label log sequences of arbitrary lengths that occur in the same file but are interleaved with benign logs.

The rule also involves a filter that limits the number of logs on which the query is executed based on their occurrence times. In the sample rule, this time range spans between start time *foothold.start* and stop time *foothold.stop* of the respective attack phase. Note that 1 second is added to the stop time to avoid incorrect label assignment due to rounding of log timestamps without sub-second precision. Filtering improves runtime performance since fewer comparisons are carried out and further decreases the probability of incorrect label assignment, e.g., benign events that match the query but are generated outside of the interval are not labeled. It is also possible to filter logs based on value matches or already assigned labels.

Sub Query Rule. Some logs do not contain all fields required to assign labels to them. Instead, it is necessary to link them to other events to determine whether they correspond to attacker behavior or not. Therefore, we propose sub query rules that involve a two-stage query mechanism: First, a main query is executed to retrieve a set of events. Then, each of these events is used in another query that allows matching based on the fields of the main query result in addition to the usual matching based on facts.

Consider the example in Fig. 4.6 which aims to label attacker requests in Apache access logs that cannot be labeled by the corresponding responses through the sequence rule from Fig. 4.5, e.g., because the response is not sent or lost. The main query retrieves

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

```
- type: elasticsearch.sub_query
  id: attacker.foothold.apache.access_dropped
  labels: [attacker_http, foothold]
  index: [pcap-attacker_0]
  query:
    - term:
        destination.ip: "{{ servers.intranet_server.default_ipv4_address }}"
  sub_query:
    index: [apache_access-intranet_server]
    query:
      - term:
          url.full: "{{ HIT.url.full }}"
      - term:
          source.address: "{{ attacker.vpn_ipv4_address }}"
```

Figure 4.6: Labeling rule of sub query type.

all log events with destination IP addresses matching the IP of the intranet server from PCAP logs in index *pcap-attacker_0*. The sub query then iterates over each of these events and labels all Apache access logs from index *apache_access-intranet_server* that have the attacker's VPN IP in field *source.address* and the same *url.full* as the PCAP event that is accessible through variable *HIT*.

We point out that sub queries have the disadvantage of a long runtime since a new query needs to be executed for each result of the main query. Accordingly, it is usually necessary to include time- and attribute-based filters in the main query to keep the number of retrieved logs to a manageable size. We omit these filters in the sample for brevity and refer to our open-source implementation.

Parent Query Rule

Parent query rules function similar to sub query rules, i.e., they comprise a main query and execute another nested query for each of the retrieved lines. However, while sub query rules label the results of the nested query, the purpose of parent query rules is to assign labels to each retrieved log of the main query if the nested parent query yields at least n results, where $n = 1$ by default. This rule type is especially useful to assign labels to special log events that were missed by earlier executed rules, e.g., timeout events that occur some time after the recorded stop time of the respective attack. In this case, the main query selects all logs within an extended time interval (i.e., a sufficiently long delta is added to the stop time in the filter) that do not have a specific label assigned, and the parent query then iterates over all results and queries for related events with matching attributes that ascertain whether the event in question is actually part of the attack. In case that at least one related event is retrieved in the parent query, a label is assigned to the respective log from the main query.

Figure 4.7 shows a parent query rule for labeling dropped DNS logs corresponding to query retries that occur when the DNS server is inactive and the malicious process

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
- type: elasticsearch.parent_query
  id: dnsteal.domain.dropped-retry
  labels: [dnsteal-dropped]
  index: [dnsmasq-inet-firewall]
  filter:
    range:
      "@timestamp":
        gt: "{{ attacker.exfiltration_service.stop }}"
        lte: "{{ (attacker.exfiltration_service.stop | as_datetime) +
            timedelta(minutes=3) }}"
  query:
    match:
      labels.rules: dnsteal.domain.match
  parent_query:
    index: [dnsmasq-inet-firewall]
    query:
      bool:
        must:
          - term:
              dns.question.name: "{{ HIT.dns.question.name }}"
          - term:
              event.action: "{{ HIT.event.action }}"
          - term:
              source.ip:  "{{ HIT.source.ip }}"
```

Figure 4.7: Labeling rule of parent query type.

generating the queries has already stopped. The main query therefore selects all logs labeled *dnsteal.domain.match* by a previously executed query rule that matches the attacker's domain name and filters them for the time range between the end of the attack execution and three minutes thereafter. The parent query then checks whether there exists an earlier occurring log event with matching query, event type, and IP, in which case the respective log line from the main query is assumed to represent a retry of an event generated by the attacker and labeled as such. Note that parent query rules have similar disadvantages with respect to the runtime as sub query rules and thus also benefit from filters. In particular, we use a time range filter to limit the results of the parent query, but omit it in the example for brevity.

Rule Selection

The previous sections outlined four types of labeling rules. Thereby, each type offers specific functionalities for labeling logs in certain situations where common labeling strategies cannot be applied. To ease the selection of appropriate rule types depending on the log data at hand, we outline a procedure that maps properties of attack artifacts to the available types.

Figure 4.8 depicts this procedure as a flow chart. Whenever it is possible to limit the queried logs to a certain time interval, e.g., the start and stop times of attacks, we recommend to add time-based filters. Second, the presence of certain attributes or labels

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

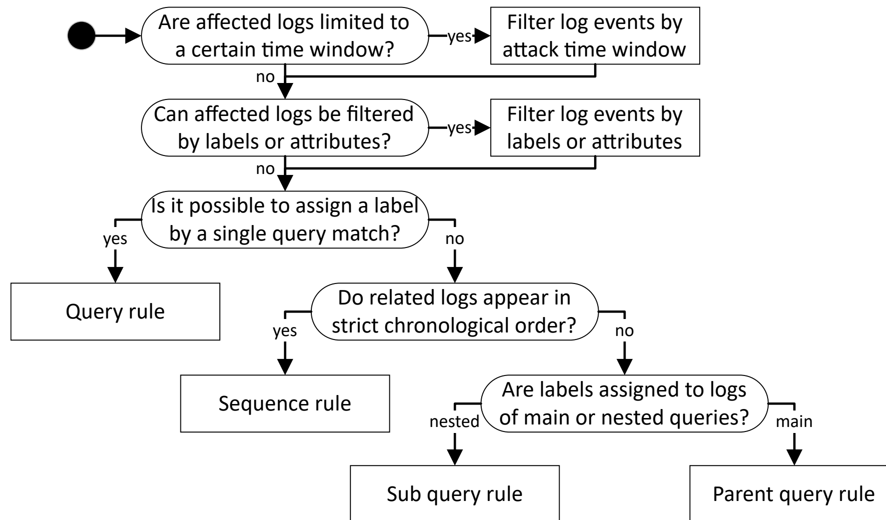


Figure 4.8: Procedure for labeling rule type selection.

assigned by previously executed rules allow to further reduce the number of logs. Query rules assign labels to logs retrieved by single queries. In case that attacks reflect in chronological sequences or correlating events across files, sequence rules should be applied. Otherwise, sub query rules can be used to label logs retrieved by nested queries and parent query rules can be used to label logs retrieved by main queries that also fulfill constraints from nested queries.

Implementation

This section summarizes our implementation decisions of aforementioned concepts. We realize testbed deployment as well as data collection with Ansible [Ans] roles. All processors of the labeling layer are implemented as scripts. We use open-source Logstash [SE19, Log] parsers that are available for a large number of common log formats and integrate well with Elasticsearch [SE19, Ela], which we use as a database for log storage. The main advantage of Elasticsearch for our approach is that it is designed for carrying out complex queries on semi-structured data efficiently. We define our rule types in YAML syntax based on the Elasticsearch query language and use the Event Query Language [EQL] for sequence rules. Finally, we generate rules of these types as templates using the Jinja templating engine [Jin].

4.3 Analysis of Log Datasets

The previous section outlined our methodology and scenario for generating testbeds using a model-driven approach. Following this methodology, we generated eight testbeds and collected log data from them. This section provides some insights into these datasets by analyzing and comparing the logs.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3.1 Testbed Infrastructures

In course of around four weeks we instantiated a total of eight testbeds that we used to collect log datasets. The durations of the simulations for each dataset are between 4-6 days, where the exfiltration attack that is already running in the beginning of the simulation usually stops after 1-3 days and the multi-step server takeover attack usually takes place on one of the last two days.

Table 4.5 provides an overview of the technical infrastructure used to generate each of the datasets. Note that we refer to each dataset by the randomly selected name of the overall testbed network that contains all zones. As visible in the table, the randomly selected numbers of mail servers and user host machines present in the testbeds correspond to the parameter variations stated in Sect. 4.2.1. We point out that the size of the datasets mostly depend on the number of active users and the length of the simulation.

Table 4.6 shows which log files are collected from which hosts, where ✓indicates that the respective log file is collected from the host, ✅ indicates that the respective log file is collected and also labels exist for that file, and no symbol indicates that the respective files are not collected or not present on the hosts. The table also shows that we collect network traffic as well as system logs from diverse sources, for example, access logs, low-level logs of the operating system (audit logs), application logs (Horde and VPN logs), monitoring logs, custom logs for state machine executions, etc. Note that files not marked as labeled do not necessarily lack a ground truth, since several files are not affected by any of the attacks and thus all occurring events correspond to normal behavior. We therefore only mark files as labeled in case that attack traces are known to occur in these files and labeling rules for the respective attack manifestations exist.

As visible in the table, we mainly focused on log files from the intranet server when developing our labeling rules. The reason for this is that the majority of attack steps are launched against that server and the diversity of these attack vectors cause that several different files are affected. In Sect. 4.3.4 we provide a more detailed overview of assigned labels.

4.3.2 Normal Behavior

It is essential for synthetic log data generation to simulate normal user behavior that corresponds to real humans interacting with the system in terms of click frequency as well as complexity and diversity of actions. However, we noticed in our literature review (cf. Sect. 2.1) that comparisons of presented datasets with real user behavior are rarely carried out. We therefore validate our log datasets by carrying out a comparison with real-world log data generated by humans performing tasks in a similar network environment. The real log data was collected during a cyber security exercise [Pla21] that took place in September of 2021. As part of the exercise, eight teams consisting of four people respectively were tasked to investigate traces of existing malware that infected their networks, monitor their systems for incoming cyber attacks, and respond to incidents by contacting authorities. As part of this one-day exercise, several attacks were

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

Table 4.5: Technical infrastructure of testbeds

Dataset	Network	Mail servers	Internal employees	Remote employees	External users	Start	End	Duration
fox	fox.org	4	5	4	7	2022-01-15 00:00	2022-01-20 00:00	5 days
harrison	harrison.com	2	3	6	6	2022-02-04 00:00	2022-02-09 00:00	5 days
russellmitchell	russellmitchell.com	2	4	3	3	2022-01-21 00:00	2022-01-25 00:00	4 days
santos	santos.com	2	9	3	6	2022-01-14 00:00	2022-01-18 00:00	4 days
shaw	shaw.info	3	5	5	3	2022-01-25 00:00	2022-01-31 00:00	6 days
wardbeck	wardbeck.info	3	6	7	4	2022-01-19 00:00	2022-01-24 00:00	5 days
wheeler	wheeler.biz	4	8	6	8	2022-01-26 00:00	2022-01-31 00:00	5 days
wilson	wilson.com	2	7	8	9	2022-02-03 00:00	2022-02-09 00:00	6 days

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 4.6: Log files collected from hosts

Host	Statemachine logs	Network traffic	Apache access logs	Apache error logs	Auth logs	Journal logs	DNS logs	VPN logs	Syslog	Audit logs	Suricata event logs	Suricata fast logs	Suricata stats	Kernel logs	Exim logs	Horde access logs	Horde error logs	Mail (info) logs	Mail warning logs	Messages	User logs	Monitoring logs
attacker host	✓	✓																				
employee host	✓																					
intranet server		✓	(✓)	(✓)	(✓)	✓			✓	(✓)	✓	✓	✓									(✓)
file share		✓	✓	✓	✓	✓			✓	(✓)	✓	✓	✓									✓
int. mail server		✓	✓	✓	✓				✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	
ext. mail server					✓				✓						✓	✓	✓	✓	✓	✓	✓	
firewall		✓			✓	✓	(✓)		✓	✓	✓	✓	✓	✓								
DNS server					✓	✓	✓		✓													
VPN server		✓			✓	✓		(✓)	✓	✓	✓	✓	✓									
web server		✓			✓	✓			✓	✓	✓	✓	✓									
cloud share		✓			✓	✓			✓	✓	✓	✓	✓									

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

scheduled for automatic execution at specific points in time, keeping the participants busy at all times. During the exercise, the teams worked isolated from each other and could not access the technical infrastructure of other teams.

To set up the system environment for each team, most of the provisioning scripts were reused as TIMs for setting up the testbed as outlined in Sect. 4.2.1. This allows us to compare the contents of the log files generated in the environments utilized by real humans and those of our dataset. We select the DNS logs as a base for comparison, since they contain queries on a level of abstraction that allows us to determine whether users accessed the cloud server, mail server, file share, etc. Figure 4.9 visualizes the events produced of the real users (left) and simulated users (right). Note that we only use logs from the first day of each dataset since there is also just one day of logs from real users available.

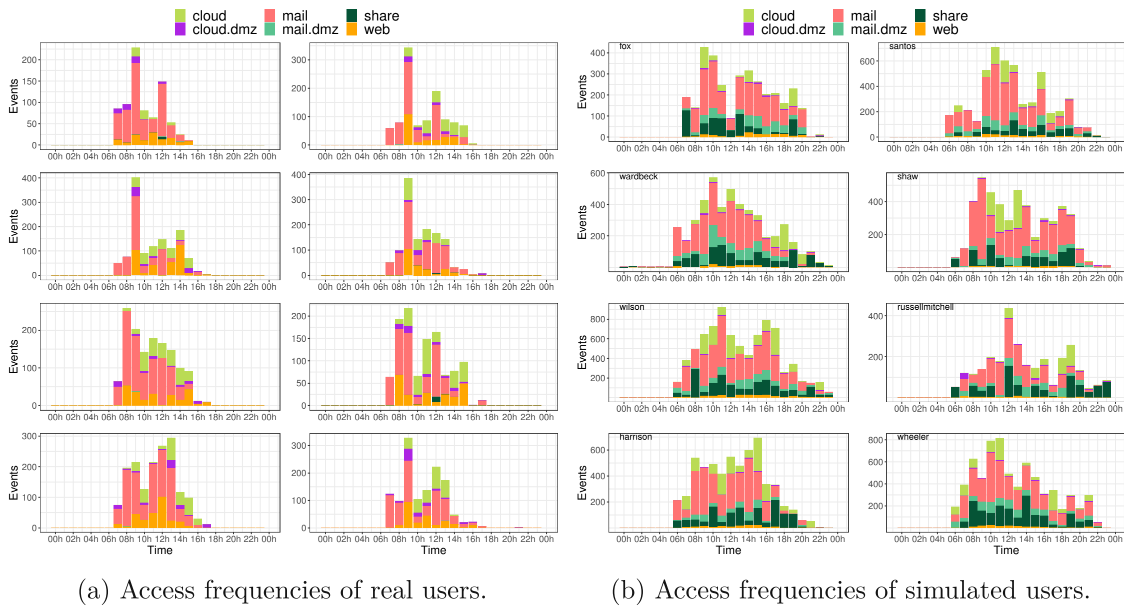
The plots show that there are some discrepancies between real and simulated users, however, these are mostly linked to some conscious design decisions. First, it is apparent that logs generated by simulated users are more spread out across the day with logs occurring between 5:00-22:00, while real users only produced logs between 7:00-17:00. This is clearly caused by the fact that the cyber security exercise had a clear start and end time and participants were not freely able to carry out their tasks at any time they desire. Accordingly, we argue that the user behavior in our datasets that simulates employees rather than participants of an exercise adequately represents the active times of employees with flexible working hours. Similarly, real logs show that users hardly ever accessed the file share, which is mostly due to the fact that none of their tasks were linked to sharing files with each other. Overall, the relative frequencies of accesses per service from real users largely resemble those of simulated users, with mail servers being the most actively accessed services. Considering the absolute event frequencies, the simulation appears to correctly depict access frequencies of real users in terms of average accesses per person and hour as well as fluctuations thereof across the day. In particular, we computed that real users generate 306.2 DNS events per day across all services on average with a standard deviation of 62.2 and simulated users generate 307.2 DNS events per day across all services on average with a standard deviation of 56.7.

4.3.3 Attacks

Manifestations of attack executions in log data and labels thereof are crucial for log datasets. As discussed in Sect. 4.2.3, we designed our attack scenario to involve a wide variety of attack types that affect several different files. In the following, we exemplarily show how some of these attack steps manifest themselves in the generated datasets.

One of the most recognizable attack steps is the directory scan that is carried out as part of the reconnaissance phase. This attack makes several thousands of requests in a short amount of time to the targeted web server, of which all are recorded in the Apache access logs. Since this log file usually contains events that relate to users requesting resources by clicking around on web pages, the scan causes a drastic increase of the average load

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



(a) Access frequencies of real users.

(b) Access frequencies of simulated users.

Figure 4.9: Event counts in DNS logs for different services.

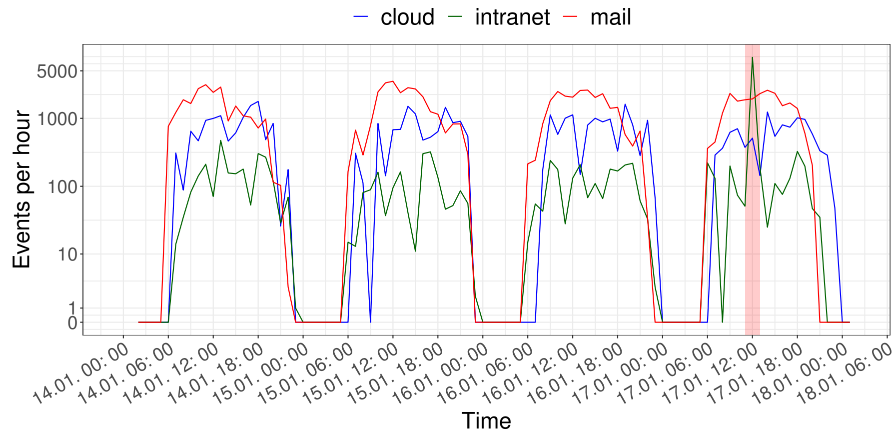


Figure 4.10: Apache access logs with attack consequences of scans.

during normal system operation. Figure 4.10 shows the number of events in the Apache access logs per hour on the cloud, intranet, and mail servers of the *santos* dataset. As visible in the plot, the accesses on the intranet server during the directory scan (the relevant time interval is shaded red) increase from several hundred to more than 5000.

Monitoring logs contain numeric values of system measurements that are an interesting input for anomaly detection [KAW11]. This includes measurements on the utilization of CPU, memory, disk, file system, network communication, processes, etc. For our datasets, we collect such monitoring logs from the file share and intranet server that

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

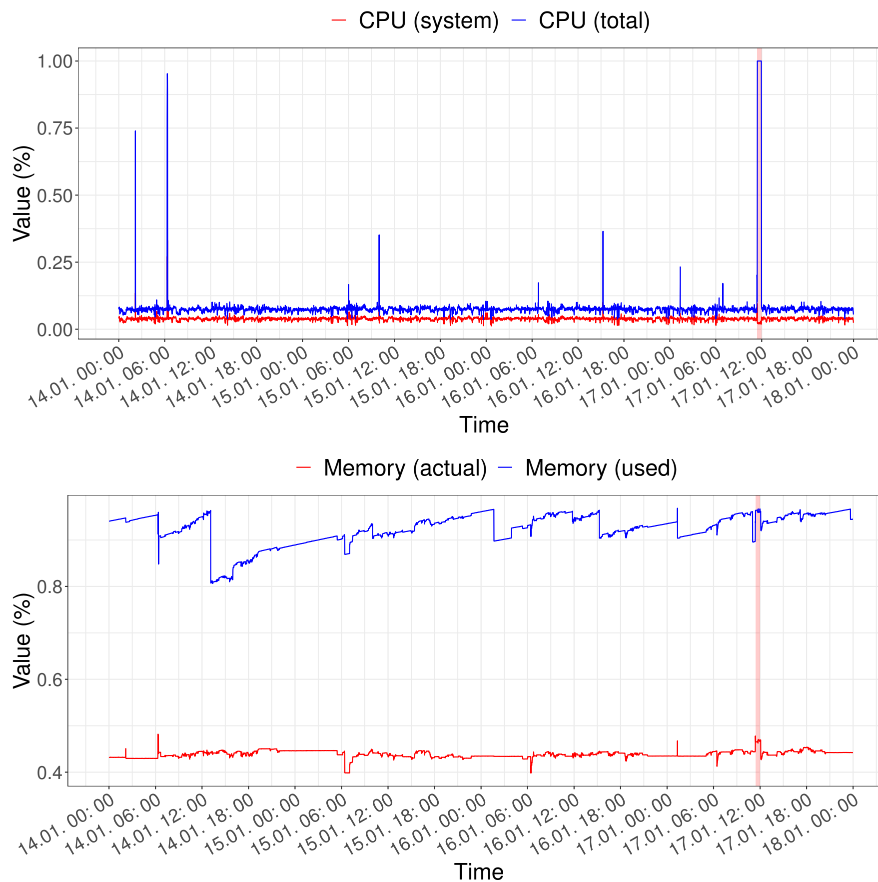


Figure 4.11: Monitoring logs of CPU (top) and memory (bottom) showing attack consequences of password cracking.

are both located in the intranet zone and are thus reasonable targets for monitoring in real-world scenarios. Figure 4.11 shows several metrics derived from CPU and memory utilization that are collected from the *santos* dataset. As visible in the top plot, both system and total CPU are significantly increased as a consequence of the password cracking attack step (the relevant time interval is shaded red). The memory metrics do not show such a strong indication of an ongoing attack, even though a large file containing passwords is loaded into memory during cracking. Nonetheless, these and other metrics or combinations thereof could also contribute to the detection of certain attack steps.

Variations of the system environment, normal behavior simulation, and attack parameters, cause that aforementioned attack consequences differ across datasets. For example, peaks in event frequencies have different magnitudes relative to the baseline of event occurrences that is considered normal for that dataset, and the time intervals where system metrics are affected change in length. In addition, event sequences that are generated as a consequence of commands executed by the attacker have different form or parameters.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
Jan 18 13:14:31 intranet-server su[28816]: Successful su for phopkins by www-data
Jan 18 13:14:31 intranet-server su[28816]: + /dev/pts/1 www-data:phopkins
Jan 18 13:14:31 intranet-server su[28816]: pam_unix(su:session): session opened for
  user phopkins by (uid=33)
Jan 18 13:14:31 intranet-server systemd-logind[1011]: New session c1 of user phopkins.
Jan 18 13:14:31 intranet-server systemd: pam_unix(systemd-user:session): session opened
  for user phopkins by (uid=0)
Jan 18 13:14:41 intranet-server sudo: phopkins : TTY=pts/1 ; USER=root ; COMMAND=list
Jan 18 13:14:43 intranet-server sudo: phopkins : TTY=pts/1 ; USER=root ;
  COMMAND=/bin/ls -laR /root/

Feb  8 08:36:38 intranet-server su[28321]: Successful su for jward by www-data
Feb  8 08:36:38 intranet-server su[28321]: + /dev/pts/0 www-data:jward
Feb  8 08:36:38 intranet-server su[28321]: pam_unix(su:session): session opened for
  user jward by (uid=33)
Feb  8 08:36:38 intranet-server systemd-logind[935]: New session c1 of user jward.
Feb  8 08:36:38 intranet-server systemd: pam_unix(systemd-user:session): session opened
  for user jward by (uid=0)
Feb  8 08:36:54 intranet-server sudo:    jward : TTY=pts/0 ; USER=root ; COMMAND=list
Feb  8 08:36:57 intranet-server sudo:    jward : TTY=pts/0 ; USER=root ;
  COMMAND=/bin/cat /etc/shadow
```

Figure 4.12: Different log events caused by the attacker escalating to system privileges in the *fox* (top) and *harrison* (bottom) datasets.

Consider the log events shown in Fig. 4.12 as an example. In the *fox* dataset (top), seven events are generated when the attacker logs into the compromised user account *phopkins*. The same attack step appears different in the *harrison* dataset, as both the affected user changes to *jward*, terminal */dev/pts/0* rather than */dev/pts/1* is used, and different commands are executed. We argue that these variations are useful to achieve higher robustness of results when evaluating IDSs, since detection accuracy should be similar across all datasets even though the events to be detected vary.

4.3.4 Labels

As explained in Sect. 4.2.4, our labeling procedure does not just make use of attack time windows to mark events as malicious based on their timestamps, but instead involves query rules that enable labeling based on event attributes. We created such rules for eight files as outlined in Table 4.6 and assign distinct labels to malicious events based on their attack step. Note that we specifically selected files and attack steps which involve distinct manifestations of attack consequences after manually checking all files, however, we also point out that there are traces of attack steps in other files that are not labeled. Due to the fact that our collection of log datasets is maintainable and the labeling procedure is repeatable, it is possible to add labeling rules for these files in future versions of the dataset.

We exemplarily show an overview of labeled events related to the multi-step attack of the *santos* dataset in Fig. 4.13. The figure visualizes the chronological occurrence of labeled events, where the distinct labels are depicted on the vertical axis and affected files are marked with different symbols. As visible in the plot, some attack steps cause

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

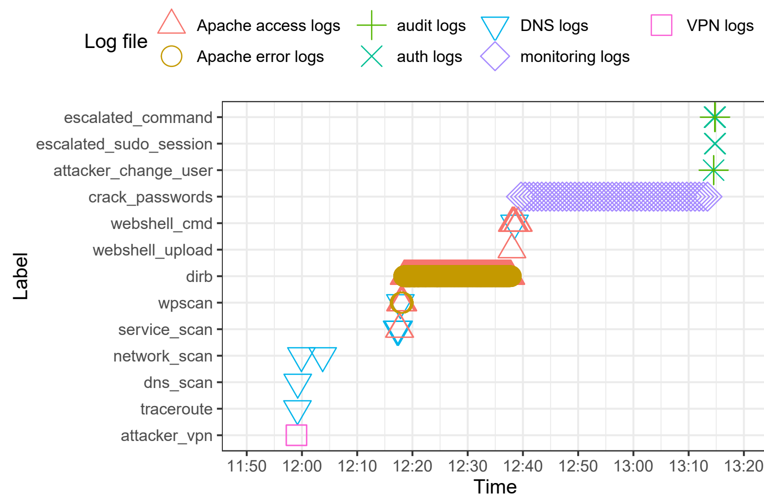


Figure 4.13: Occurrences of events labeled as part of the attack steps.

singular events or short sequences (e.g., uploading the webshell), while others affect groups of events that span over a longer duration (e.g., password cracking). Note that we assign multiple labels to the same events for clarification. For example, we introduce a label *foothold* that subsumes all attack steps involved in the initial intrusion, including the VPN connection, scans, and webshell upload. This implies that our labels follow a hierarchical order, which makes it easy to select specific types of events for evaluation and furthermore allows to compute detection accuracies separately for different attack steps.

Table 4.7 provides an overview of the numbers and types of rule templates and their corresponding labels. For example, the template rule shown in Fig. 4.4 is a query rule that assigns label *dnsteal* and thus contributes to the respective counter in column *Query*. Note that it is possible that a single rule assigns multiple labels and that different rules assign the same label. For this reason, the total number of rules displayed in the last row does not represent the sum of all labels assigned per rule type, but instead the total number of rules independent from the number of labels they assign. For example, we designed three parent query rule templates and two of them assign labels *foothold* and *attacker_http*, thus both counters for these labels show 2. Overall, this breakdown shows that the majority of rules necessary for our scenario comprise query and sequence rules, while comparatively few sub query and parent query rules are required.

Table 4.8 depicts the number of labeled log events in each file in the *santos* dataset, where the top header row states the host at which the file was collected, and the bottom header row states the specific file and its number of lines in brackets. Similar to rules, lines can be assigned multiple labels and thus the bottom row depicts the total number of labeled events per file. However, the total number of assigned labels is depicted across all files in the column on the right hand side. The results indicate that the numbers of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Table 4.7: Overview of the number of rule templates for each log data label

	Rule templates			
Label	**Query**	**Seq.**	**Sub**	**Parent**
dnsteal	2			
dnsteal-received	1			
dnsteal-dropped	1			1
exfiltration-service	1			
foothold	9	5	7	2
attacker_http		4	3	2
dirb	2		1	
wpscan	2			
service_scan	2		1	
dns_scan	2		1	
network_scan	1		1	
traceroute	1			
webshell_upload		1		
webshell_cmd	3			
escalate	4	12		
crack_passwords	1			
attacker_change_user		6		
escalated_command		5		
escalated_sudo_session		1		
attacker_vpn	2	2		
Total	20	18	7	3

labeled events differ greatly depending on the attack and considered file, i.e., most labels are related to the data exfiltration that mainly affects the "firewall/dnsmasq.log" file.

Figure 4.14 depicts an Euler diagram of labeled events. The visualization makes it easy to see the hierarchical structure of labels, e.g., the dirb attack involves HTTP traffic and thus logs labeled *dirb* are a subset of logs labeled *attacker_http*.

We point out that due to the variations of the testbeds and attack parameters, the resulting numbers of labeled events vary greatly. For example, the configurations of the directory scan in the *fox* dataset cause that 406045 rather than only 4462 events are generated in the access logs. On the other hand, the *shaw* dataset does not involve traces of the data exfiltration attack in the DNS logs due to specific settings of DNSteal (in particular, IP forcing is activated). In the following section, we will evaluate the correctness of these label assignments for one particular dataset.

Expert Survey

The previous section provided an overview of the labels assigned to attack traces. We evaluate our approach by validating the correctness of these labels. For this, we set up a survey and ask security engineers and IT analysts to review the logs and labels. Note

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 4.8: Overview of the number of labeled lines per file for each log data label

	vpn	intranet				monitoring	share	firewall	
Label	**openvpn.log** (5622)	**wp-access.log** (15784)	**wp-error.log** (46)	**auth.log** (612)	**audit.log** (2263)	**system.cpu.log** (7680)	**audit.log** (728)	**dnsmasq.log** (275667)	**Total**
dnsteal							2	38561	38563
dnsteal-received								38651	38561
exfiltration-service							2		2
foothold	28	7790	42						7860
attacker_http		7786	42						7828
dirb		4462	23						4485
wpscan		3285	19					8	3312
service_scan		12						315	327
dns_scan								414	414
network_scan								112	112
traceroute								4	4
webshell_upload		3							3
webshell_cmd		32						12	44
escalate		4		12	19	44		12	91
crack_passwords						44			44
attacker_change_user				5	9				14
escalated_command				8	10				18
escalated_sudo_session				6					6
attacker_vpn	28								28
Total	28	7794	42	12	19	44	2	39426	47367

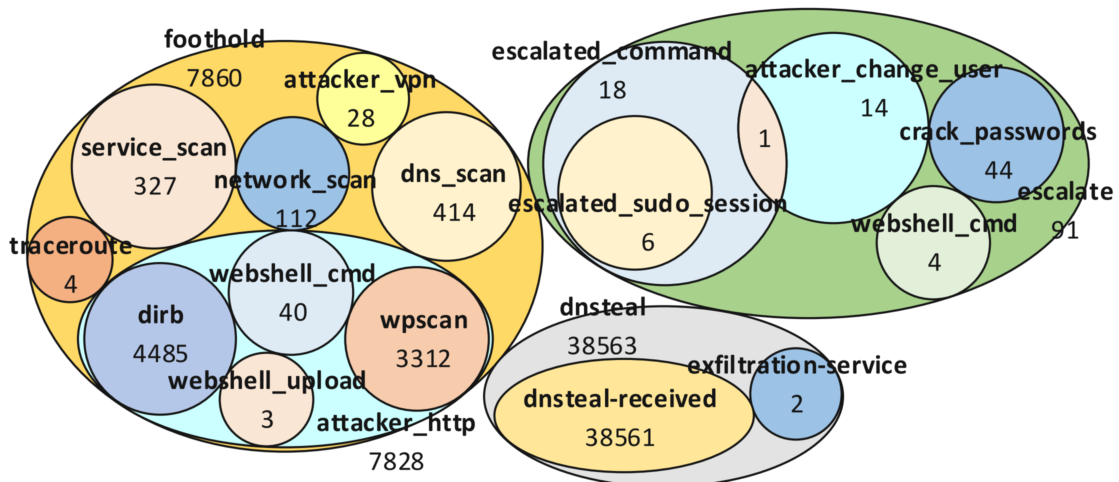


Figure 4.14: Euler diagram depicting label relationships.

that the labeled dataset used as a basis for the survey is not one of the eight published datasets, but a preliminary dataset [LFS+21] with fewer attack cases that was only generated to evaluate the labeling procedure and improve upon identified issues.

The used dataset involves a total of 451, 341 labels, which makes verification by humans impossible without sampling the data. We restrict the evaluation to labels *dirb*, *webshell_upload*, *webshell_cmd*, *reverse_shell*, *attacker_change_user*, *escalated_command*, *dnsteal*, and *exfiltration-service* to avoid labels of supersets so that users are presented with the most specific label. We then randomly select events corresponding to these labels and present them to survey participants. Since the context of occurrence is helpful and sometimes essential for the correct interpretation of logs, we display the events with four immediately succeeding and preceding log lines. Note that the experts also have access to the whole dataset without labels in case that they want to review more lines occurring close to the line in question. Furthermore, we provide them with relevant technical details on the testbed and the launched attacks prior to filling out the survey.

Figure 4.15 shows a sample question from the survey. The participant is informed that the *dirb* label was assigned to the marked Apache access log line and tasked to determine whether this label is correct or not. Participants may select their opinion on a seven-point scale ranging from strong disagreement to strong agreement and including "No answer" as a neutral option in case that they are unable to make a decision. The log sample in the figure depicts several requests made in a short time interval (all lines have the same timestamp) and in seemingly alphabetical order, where all requested pages start with the letters "em". A participant with sufficient technical expertise and knowledge about the attack could therefore conclude that these lines are likely artifacts of a dictionary scan, and thus agree with the assigned label *dirb*. We decided for such a quantitative evaluation over expert reviews of our developed rules for two reasons: First, we aim to ensure an objective evaluation with adequate efforts. Second, we base the evaluation on

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

55 The following are log lines taken from **scenario_1_rce/servers/intranet_server/logs/apache2/intranet.company.cyberrange.at-access.log.4** . Line **68165** (marked in red) has been labeled as **dirb**. Please decide if this label is correct or not:

```
68161 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/embed HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68162 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/embedd HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68163 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/embedded HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68164 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/emea HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68165 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/emergency HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68166 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/emoticons HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68167 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/employee HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68168 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/employees HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
68169 - 172.16.100.151 - - [23/Mar/2021:20:40:03 +0000] "GET /wp-content/themes/employers HTTP/1.1" 404 363 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

	Strongly Disagree	Disagree	Somewhat Disagree	Somewhat Agree	Agree	Strongly Agree	No answer
Line 68165 is labeled as "dirb".							●

Figure 4.15: Survey question asking the participant to decide if the marked line is correctly labeled as part of the dirb attack step.

the generated labels rather than the rules to focus on the actual output of our approach and recognize any incorrectly labeled events.

In addition to labels for attacker behavior, we also sampled unlabeled lines and asked participants to determine whether these lines actually correspond to benign behavior. Moreover, control questions ensure that participants do not just agree to all questions, but are actually able to differentiate malicious from benign behavior. We therefore add questions with purposefully incorrect labeled logs, i.e., events that received an attack label from the labeling framework but are displayed with label *normal*, and benign events that are presented with a randomly selected attack label from the same file. Overall, each survey sheet consists of 54 questions: 14 log samples correctly labeled as malicious, 14 log samples correctly labeled as benign, 12 log samples incorrectly labeled as malicious, and 14 log samples incorrectly labeled as benign. This setup allows us to identify and possibly exclude participants who select random answers or do not have the technical skills required to interpret the logs. However, we point out that the purpose of this survey is not to rate the ability of participants to recognize attacks in log data, but instead to determine whether manually assigned labels based on expert knowledge diverge from labels generated by our automatic procedure. Thereby, the survey format aims to discover incorrect rules rather than missing rules, since it is unlikely that logs without labels that are actually part of an attack are selected during sampling of benign events.

We hosted the survey online and asked engineers with security expertise for anonymous participation. In particular, we contacted cyber security experts, penetration testers, and capture-the-flag contestants and invited them to share the link to the survey among their peers. In the beginning of the survey, we asked participants about their roles. Then, the same questions were displayed in random order to each participant. After one week, we obtained responses from 16 participants, out of which 8 skipped more than 25% of all questions and were thus excluded. The remaining 8 participants skipped less than 2% of all questions on average, indicating their high confidence in filling out the survey. The majority of these participants (5) identify their roles as security analysts, 2 as penetration testers, and 1 as a cyber security research engineer. In the following, we analyze the answers of these participants.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

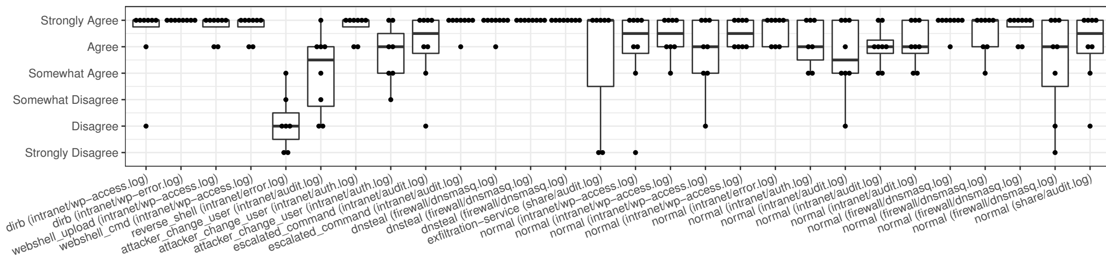


Figure 4.16: Boxplots of survey answers to correctly labeled lines show that participants mostly agreed with assigned labels.

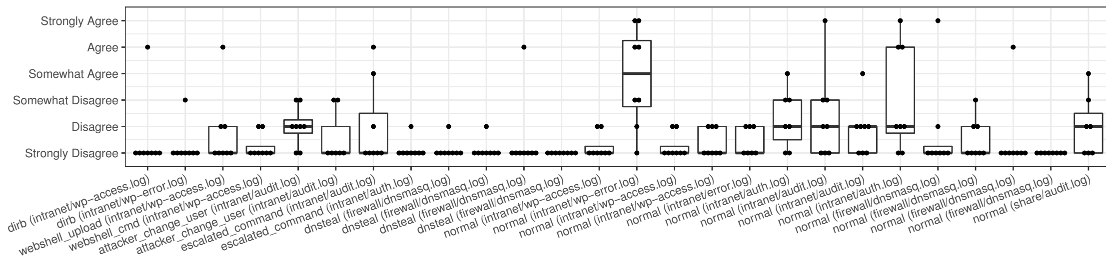


Figure 4.17: Boxplots of survey answers to incorrectly labeled lines show that participants recognized consciously placed errors.

Figure 4.16 shows the answers to questions with correct labels as a boxplot, where the labels and log files associated with the respective question are displayed on the horizontal axis and each point represents an answer. For example, the answers to the question from Fig. 4.15 are displayed on the left hand side of the plot with label *dirb (intranet/wp-access.log)* and show that 7 out of 8 participants agree or strongly agree with the assigned label. Note that "No answer" responses are excluded so that distributions are not distorted.

Overall, the plot shows a clear trend towards agreement for labeled attacks, with few exceptions. We thoroughly analyzed these outliers and ensured that the logs are in fact correctly labeled by our approach. Since the survey is anonymous, we are unable to determine the exact reasons for the disagreement of the participants, however, we see several possible explanations. For example, label *reverse_shell* in file *intranet/error.log* refers to a log event with message "Bad file descriptor". We argue that the missing timestamp and rather general error message made it difficult for participants to relate the event to the attack. Moreover, label *attacker_change_user* in file *intranet/audit.log* received mixed answers. This is likely due to two reasons: First, audit logs are designed to be machine readable rather than human readable (e.g., timestamps are in unix epoch format and values are sometimes encoded), which makes them more difficult to interpret for analysts who are not sufficiently experienced with this log format. Second, in this log sample relevant attack indicators occur in the preceding lines rather than the marked line itself and are thus more easily overlooked by participants.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Interestingly, logs correctly labeled as normal have slightly lower agreement scores on average than logs corresponding to attacks. We argue that this is due to the fact that analysts have higher confidence in their answer when they recognize indicators for specific attacks that fit the proposed label, while the lack of such indicators is not sufficient to deem normal logs as such with high certainty.

Figure 4.17 shows the survey results with respect to incorrectly labeled lines. Note that the horizontal axis shows the actual labels assigned to the logs, not the randomly selected incorrect ones. The overall trend towards disagreement indicates that participants were able to recognize incorrect labels. Similar to the previous results from the correctly labeled lines, there appears to be less consensus among participants when verifying labels of normal behavior. One extreme example is the question labeled *normal (intranet/wp-error.log)*, where logs are incorrectly labeled as *dirb* and involve "no such file or directory" warnings that are unrelated to the attacks. Since such events likely occur during scans, it is understandable that some participants were drawn towards agreement.

4.4 Discussion

In this section we discuss whether our generated datasets fulfill the requirements for IDS evaluation. In addition, we explain possible application scenarios for our datasets in detail and outline their limitations.

4.4.1 Fulfillment of Requirements

We stated requirements for log dataset generation that we used as a basis for our methodology in Sect. 2.1. Based on the generated datasets and the results of our analysis provided in Sect. 4.3 we check whether all requirements are fulfilled. Requirement (1) is fulfilled as our datasets address enterprise IT, which is a wide-spread and relevant use-case for intrusion detection. We followed common guidelines for network design and selected open-source components that are popular choices in such infrastructures [MFCMC+18]. Requirement (2) addresses simulation of normal system behavior. We argue that our state machines and randomized user role assignments that are used for simulating employees as outlined in Sect. 4.2.2 are sufficiently extensive to generate complex patterns. Moreover, we show in Sect. 4.3.2 that page visit frequencies of our simulated employees largely resemble those of real users. Similarly, our selected attack scenarios involve diverse steps and recent exploits to fulfill requirement (3). We collect both system log data as demanded by requirement (4) as well as network traffic as demanded by requirement (5). Our user simulations follow daily activity cycles as visible in our analysis results presented in Sect. 4.3.3. Since multiple days of such normal behavior is recorded, we consider requirement (6) that addresses periodic patterns as fulfilled. Requirement (7) is fulfilled as we generate a ground truth for events using our labeling framework as described in Sect. 4.2.4. Beside a description of the overall scenario in the dataset repository, all scripts and configurations of our testbeds are published together with the log data and thus also requirement (8) on the availability of documentation is fulfilled. Requirements

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

(9) and (10) are fulfilled, because we generate multiple datasets that contain repeated executions of the same attack steps with variations. Finally, requirement (11) is fulfilled as we publish all scripts for deploying and running the simulation as open-source code.

4.4.2 Application Scenarios

Due to the characteristics of our dataset, we foresee several different application scenarios. In the following, we discuss (federated) intrusion detection, alert aggregation, and user profiling as interesting research areas that benefit from our data.

Evaluation of Intrusion Detection Systems

Foremost, the purpose of our collection of datasets is to enable evaluation of host- and network-based IDSs. We injected attacks that employ diverse techniques so that their consequences manifesting in log files challenge a wide range of detection mechanisms [SWL21b]. For example, we anticipate the following non-exhaustive list of detection techniques to be applied on our dataset.

- **New log artifacts**. As part of many attack steps, new log events such as the sample logs from Fig. 4.12 appear in some log files. Alternatively, normal event types may appear with different parameters or combinations of parameter values. Despite the fact that this detection technique is relatively simple, it is highly powerful, because its low runtime requirements can be applied to most events or categorical values.

- **Structure of parameter values**. The DNSteal attack makes use of a randomly generated domain names for data exfiltration, which could be useful to evaluate detectors for domain-generation algorithms [WAAG16]. The same applies for Apache access logs, where commands sent to the webshell appear in URLs.

- **Sequence mining**. Log events usually occur in specific sequences that represent inherent program flows of monitored services. Workflow mining extracts these patterns and allows to detect unusual sequences as anomalies [DLZS17, HZHL16]. Consequences of exploits and other malicious attacker behavior often manifest in such sequences, for example, audit events generated when the attacker executes commands via the remote shell.

- **Event frequencies**. As pointed out in Sect. 4.3.3, attacks such as scans are recognizable by high amounts of log occurrences in short time intervals. Anomaly detection techniques therefore create event count matrices and detect time windows with unusual high or low event frequencies with the aid of various machine learning methods, including time-series analysis [LWS+18a, LWS+18b] and principal component analysis [HZHL16].

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

- **Missing events**. We deliberately designed our attack scenario to include a data exfiltration attack that is already ongoing at the beginning of the simulation and stops after some days. We expect that detectors based on machine learning add these malicious events to their model of normal behavior that is generated during the training phase, and thus poison their models. Accordingly, detectors need to raise anomalies for the stopping of event occurrences, which we consider a more challenging detection scenario than recognizing the start of the exfiltration process.

- **Statistical tests**. System performance metrics and numeric features of network traffic are suitable for statistical analyses such as testing for certain distributions. Alternatively, hypothesis testing is also applicable for detecting changes of correlating behavior of categorical variables in log data [LHW+21].

We argue that our data has a large benefit over most existing datasets for IDS evaluation, as it contains data from multiple separate testbeds targeted by the same attack scenario. Due to the variations in the log traces caused by changes of the system environment, simulated normal behavior, and attack parameters, we expect that detection accuracies vary when applying the same detectors on different datasets. However, by averaging the detection metrics achieved on all datasets, the aggregated results have a higher robustness as they are more representative for a general case and not fine-tuned to only a single execution [NK19]. In addition, simulating many similar infrastructures allows to evaluate approaches that leverage federated learning for intrusion detection [PRT+18].

Moreover, the ground truth tables of our datasets are not just binary labels that determine whether an event is part of an attack or not, but instead precisely state the type of attack. This means that it is also possible to evaluate attack classification accuracy in case that the detectors are capable of determining attack types, e.g., by matching them with a list of known and labeled meta-alerts.

Evaluation of Alert Aggregation Techniques

Intrusion detection techniques as stated in the previous section often raise large amounts of alerts for some attack steps, where the vast majority of these alerts are duplicates and only have little value to operators that monitor IDSs. Alert aggregation therefore attempts to merge these alerts to reduce the workload of operators and ease the identification of urgent alerts that require immediate actions. On top of that, advanced aggregation techniques are capable of recognizing patterns of alert occurrences and are able to connect attack steps to attack scenarios [LSW+19].

In order to merge alerts and attack steps, it is obviously necessary to have datasets at hand that contain repetitions of the same or similar attacks. Unfortunately, these datasets are rare even though they are urgently needed in research [NDP18]. We therefore propose to forensically analyze our datasets with a desired selection of IDSs to obtain sequences of alerts that are used for aggregation. Similar to the evaluation of IDSs, the variations of our attack scenarios come in handy as they yield different alert patterns for

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

each dataset, e.g., variable amounts of alerts for scans with varying duration or optional alerts caused by commands that the attacker only carries out with certain probabilities. This allows to evaluate whether alerts are indeed aggregated with the same attack types independent of slight variations that occur in real-world environments.

Evaluation of User Profiling Approaches

User profiling is a trending research topic that aims to create a profile for each user and then use these profiles to group users by their behavior or role. For this, algorithms based on pattern mining read out access logs that detail all page visits by each user [PMGO18]. Note that this application scenario is not related to cyber attacks, because only the simulation of normal user behavior is relevant. Due to the fact that our simulated users have specific roles (e.g., WordPress editor or administrator) and visit all pages based on transition probabilities, they clearly follow their own behavior profiles. The main advantage over real data is that it is easy to adjust these profiles according to the respective use case and to quantitatively compare their similarities, which is useful for evaluations and cannot be replicated with humans.

4.4.3 Limitations

Despite all aforementioned benefits of our log dataset, we recognize some limitations. Most important, the user simulation that generates a baseline of normal behavior for our collection of log datasets is obviously limited by the extent of our state machines. On the other hand, real datasets that contain traces of humans interacting with the monitored environments always have the possibility to involve artifacts caused by deliberate or accidental misuse of the systems that could yield incorrect alerts by IDSs. Despite our efforts to generate complex user behavior, we therefore cannot ensure that false positive rates achieved on our datasets are representative for real-world systems. Nonetheless, we are convinced that our synthetic datasets have significant advantages over real ones, as they can be freely published without the need to anonymize artifacts due to privacy concerns and may be arbitrarily recreated in modified use-cases if necessary.

We also point out that we aimed to generate the log data in the most realistic way possible, meaning that we did not configure the logging frameworks to collect data on the highest level of granularity, but instead used standard or default configurations wherever applicable. In case that logging levels need to be adapted, it is always possible to replay the attack scenarios on our open-source testbeds.

As part of varying the parameters of our testbed when generating TSMs from TIMs, we also decided to leave configurations of logging services unchanged in order to ensure that our labeling rules do not accidentally leave some events unlabeled. We leave the task of extending our labeling rules for this kind of variations for future work.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.5 Summary

In this chapter we present a collection of eight synthetic log datasets for evaluation of intrusion detection systems. We collect our datasets from testbeds generated by a model-driven methodology for testbed setup and labeling. This enables to repeat the data collection procedure arbitrary many times while at the same time varying several parameters of the simulation with low manual effort. In addition, it is simple to scale the network and extend it with additional components or services. Our datasets are openly accessible and maintainable as all code required to deploy testbeds, run simulations, and assign labels to log events is available open-source. Our datasets thus solve several problems that are prevalent in existing datasets, including control over the simulation parameters, presence of repeated attack executions in similar environments, generation of ground truth tables, complexity of the network, preprocessing of logs to protect sensitive information, and more.

Our collected log data stems from many different sources and thus involves diverse formats. This is often problematic, as IDSs and other analysis techniques for log data usually require structured data. While it is essential for realistic analysis to provide datasets with logs in raw and unprocessed format, it is just as important to aid analysts by providing suitable parsers for the data. Manual generation of such parsers is time-consuming, especially when log events have many parameters (e.g., event logs in JSON format) or involve a large number of different events (e.g., syslog). The following chapter therefore surveys log clustering algorithms in order to ease or even automatize parser generation.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 5

Log Clustering

The previous sections introduced log datasets collected from simulation testbeds. Analysis of these datasets is generally difficult as several of the collected log files contain large amounts of events. In practice, organizations typically face up to 4 million log lines per day [XHF$^+$09] and peaks of 22,000 events per second [AR19]. Even more, log datasets comprise logs from various sources, meaning that there are not only heterogeneous events within a single log file, but there are also entirely different event formats across log sources. One method to deal with this problem is to group log lines into clusters of related events, which is capable of reducing large amounts of data into coherent chunks. This facilitates manual analysis as well as generation of log parsers, i.e., rules that allow to classify events, derive their syntax, and extract parameters. This chapter therefore provides a survey on log clustering with the purpose of simple anomaly detection and identification of suitable algorithms for log parser generation. Major parts of this chapter have been published in [LSWR20].

Several clustering algorithms that were particularly designed for textual log data have been proposed in the past. Since most of the algorithms were mainly developed for certain application-specific scenarios at hand, their approaches frequently differ in their overall goals and assumptions on the input data. We were specifically interested to discover the different strategies the authors used to pursue the objectives induced by their use-cases. However, to the best of our knowledge there is no exhaustive survey on state-of-the-art log data clustering approaches that focuses on applications in cyber security. Despite also concerned with certain types of log files, existing works are either outdated or focus on network traffic classification [E$^+$13], web clustering [CORW09], and user profiling [VPD04, FL05]. Other surveys address only log parsers rather than clustering [ZHL$^+$19] or do not focus on security [HHC$^+$21].

In this chapter we therefore create a survey of current and established strategies for log clustering found in scientific literature. This survey is oriented towards the identification of overall trends and highlights the contrasts between existing approaches. This supports

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

analysts in selecting methods that fit the requirements imposed by their systems. In addition, we aim at the generation of a work of reference that is helpful for all authors planning to publish in this field. Overall, the research questions we address in this chapter are as follows:

- What are essential properties of existing log clustering algorithms?
- How are these algorithms applied in cyber security?
- On what kind of data do these algorithms operate?
- How were these algorithms evaluated?

The remainder of the chapter is structured as follows. Section 5.1 outlines the problem of clustering log data and discusses how log analysis is used in the cyber security domain. In Sect. 5.2, we explain our method of carrying out the literature study. The results of the survey are stated and discussed in Sect. 5.3. Finally, Sect. 5.4 summarizes the chapter.

5.1 Survey Background

Log data exhibits certain characteristics that have to be taken into account when designing a clustering algorithm. In the following, we therefore discuss important properties of log data, outline the reasons why log data is suitable to be clustered and look into application scenarios relevant to cyber security.

5.1.1 The Nature of Log Data

Despite the fact that log data exists in various forms, some general assumptions on their compositions can be made. First, a log file typically consists of a set of single- or multi-line strings listed in inherent chronological order. This chronological order is usually underpinned by a time stamp attached to the log messages[1]. The messages may be highly structured (e.g., a list of comma-separated values), partially structured (e.g., attribute-value pairs), unstructured (e.g., free text of arbitrary length) or a combination thereof. In addition, log messages sometimes include process IDs (PIDs) that relate to the task (also referred to as thread or case) that generated them. If this is the case, it is simple to extract log traces, i.e., sequences of related log lines, and perform workflow and process mining [NMA+16]. Other artifacts sometimes included in log messages are line numbers, an indicator for the level or severity of the message (TRACE, DEBUG,

[1]The order and time stamps of messages do not necessarily have to correctly represent the actual generation of log lines due to technological restrictions appearing during log collection, e.g., delays caused by buffering or issues with time synchronization. A thorough investigation of any adverse consequences evoked by such effects is considered out of scope for this thesis.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

INFO, WARN, ERROR, FATAL, ALL, or OFF), or a static identifier referencing the statement printing the message [BLL+18].

Arguably, log files are fairly different from documents written in natural language. This is not necessarily the case because the log messages themselves are different from natural language (since they are supposed to be human-readable), but rather because of two reasons: (i) Similar messages repeat over and over. This is caused by the fact that events are recurring since procedures are usually executed in loops and the majority of the log lines are generated by a limited set of print statements, i.e., predefined functions in the code that write formatted strings to some output. (ii) The appearances of some messages are highly correlated. This is due to the fact that programs usually follow certain control flows and components that generate log lines are linked with each other. For example, two consecutive print statements will always produce perfectly correlated log messages during normal system behavior since the execution of the first statement will always be followed by the execution of the second statement. In practice, it is difficult to derive such correlations since they often depend on external events and are the result of states and conditions.

These properties allow system logs to be clustered in two different ways. First, clustering individual log lines by the similarity of their messages yields an overview of all events that occur in the system. Second, clustering sequences of log messages gives insight into the underlying program logic and uncovers otherwise hidden dependencies of events and components.

5.1.2 Static Clustering

We consider clustering individual log lines as a static procedure, because the order and dependencies between lines is usually neglected. After such static line-based clustering, the resulting set of clusters should ideally resemble the set of all log-generating print statements, where each log line should be allocated to the cluster representing the statement it was generated by. Examining these statements in more detail shows that they usually comprise static strings that are identical in all messages produced by that statement and variable parts that are dynamically replaced at run time. Thereby, variable parts are frequently numeric values, identifiers (e.g., IDs, names, or IP addresses), or categorical attributes. Note that the generation of logs using mostly fixed statements is responsible for a skewed word distribution in log files, where few words from the static parts appear very frequently while the majority of words appears very infrequently or even just once [Vaa03, NJCY14].

In the following, we demonstrate issues in clustering with the sample log lines shown in Fig. 5.1. In the example, log messages describe users logging in and out. Given this short log file, a human would most probably assume that the two statements `print("User " + name + " logs in with status " + status)` and `print("User " + name + " logs out with status " + status)` generated the lines, and thus allocate lines $\{1, 2, 4\}$ to the former and lines $\{3, 5\}$ to the latter cluster. From this clustering,

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
1 :: User Alice logs in with status 1
2 :: User Bob logs in with status 1
3 :: User Alice logs out with status 1
4 :: User Charlie logs in with status -1
5 :: User Bob logs out with status 1
```

Figure 5.1: Sample log messages for static analysis.

the templates (also referred to as signatures, patterns, or events) “User * logs in with status *” and “User * logs out with status *” can be derived, where the Kleene star * denotes a wildcard accepting any word at that position. Beside the high resemblance of the original statements, the wildcards appear to be reasonably placed since all other users logging in or out with any status will be correctly allocated, e.g., “User Dave logs in with status 0”.

Other than humans, algorithms lack semantic understanding of the log messages and might just as well group the lines according to the user name, i.e., create clusters $\{1, 3\}$, $\{2, 5\}$, and $\{4\}$, or according to a state variable, i.e., create clusters $\{1, 2, 3, 5\}$ and $\{4\}$. In the latter case, the most specific templates corresponding to the clusters are “User * logs * with status 1” and “User Charlie logs in with status -1”. In most scenarios, the quality of these templates is considered to be poor, since the second wildcard of the first template is an over-generalization of a categorical attribute and the second template is overly specific. Accordingly, newly arriving log lines would be likely to form outliers, i.e., not match any cluster template.

With this example in mind we want to point out that there always exist a multitude of different possible valid clusterings and judging the quality of the clusters is eventually a subjective decision that is largely application-specific. For example, investigations regarding user-behavior may require that all log lines generated by a specific user end up in the same cluster. In any way, appropriate cluster quality is highly important since clusters are often the basis for further analyses that operate on top of the grouped data and extracted templates. The next section explores dynamic clustering as such an application that utilizes static cluster allocations.

5.1.3 Dynamic Clustering

As pointed out earlier, log files are suited for dynamic clustering, i.e., allocation of sequences of log line appearances to patterns. However, raw log lines are usually not suited for such sequential pattern recognition, due to the fact that each log line is a uniquely occurring instance describing a part of the system state at a particular point in time. Since pattern recognition relies on repeating behavior, the log lines first have to be allocated to classes that refer to their originating event. This task is enabled by static clustering as outlined in the previous section.

In the following, we consider the sample log file shown in Fig. 5.2 that contains three users logging into the system, performing some action, and logging out. We assume that

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
1 :: User Alice logs in with status 1    :: A
2 :: User Alice performs action open     :: B
3 :: User Alice logs out with status 1   :: C
4 :: User Bob logs in with status 1      :: A
5 :: User Bob performs action write      :: B
6 :: User Charlie logs in with status 1  :: A
7 :: User Bob logs out with status 1     :: C
8 :: User Charlie performs action exec   :: B
9 :: User Charlie logs out with status 1 :: C
```

Figure 5.2: Sample log messages and their event allocations for dynamic analysis.

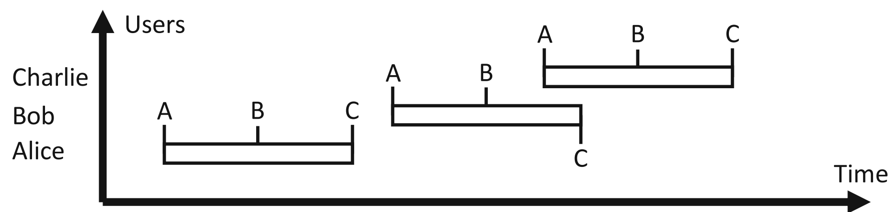


Figure 5.3: Sample log events visualized on a timeline.

these steps are always carried out in this sequence, i.e., it is not possible to perform an action or log out without first being logged in.

We assume that the sample log file has been analyzed by a static clustering algorithm to generate the three templates A="User * logs in with status *", B="User * performs action *", and C="User * logs out with status *". It is then possible to assign each line one of the events as indicated on the right side of the figure. In such a setting, the result of a dynamic clustering algorithm could be the extracted sequence A, B, C since this pattern describes normal user behavior. However, the events in lines 6 and 7 are switched, thus interrupting the pattern. Figure 5.3 shows that the reason for this issue is caused by interleaved user behavior, i.e., user Charlie logs in before user Bob logs out.

Since many applications are running in parallel in real systems, interleaved processes are commonly occurring in log files and thus complicate the pattern extraction process. As mentioned in Sect. 5.1.1, some log files include process IDs that allow to analyze the corresponding logs isolated from interrupting processes and thus resolve this issue. In the simple example from Fig. 5.2, the username could have been used for this purpose. In addition to interleaved event sequences, real systems obviously involve much more complex patterns, including arbitrarily repeating, optional, alternative, or nested subpatterns.

While sequence mining is common, it is not the only dynamic clustering technique. In particular, similar groups of log lines can be formed by aggregating them in time-windows and analyzing their frequencies, co-occurrences, or correlations. For example, clustering could aim at generating groups of log lines that frequently occur together. Note that in this setting, the ordering of events is not relevant, but only their occurrence within a

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

certain time interval. The next section outlines several applications of static and dynamic clustering for system security.

5.1.4 Applications in the Security Domain

Due to the fact that log files contain permanent documentation of almost all events that take place in a system, they are frequently used by analysts to investigate unexpected or faulty system behavior in order to find its origin. In some cases, the strange behavior is caused by system intrusions, cyber attacks, malware, or any other adversarial processes. Since such attacks often lead to high costs for affected organizations, timely detection and clarification of consequences is of particular importance.

Independent from whether anomalous log manifestations are caused by randomly occurring failures or targeted adversarial activity, their detection is of great help for administrators and may prevent or reduce costs. Clustering is able to largely reduce the effort required to manually analyze log files, for example, by providing summaries of log file contents, and even provides functionalities to automatize detection of anomalous behavior. In the following, we outline some of the most relevant types of anomalies detectable or supported by clustering.

- **Outliers** are single log lines that do not match any of the existing templates or are dissimilar to all identified clusters that are known to represent normal system behavior. Outliers are often new events that have not occurred during clustering or contain highly dissimilar parameters in the log messages. An example could be an error log message in a log file that usually only contains informational and debugging messages.
- **Frequency anomalies** are log events that appear unexpectedly frequent or rare during a given time interval. This may include cases where components stop logging, or detection of attacks that involve the execution of many events, e.g., vulnerability scans.
- **Correlation anomalies** are log events that are expected to occur in pairs or groups but fail to do so. This may include simple co-occurrence anomalies, i.e., two or more events that are expected to occur together, and implication anomalies, where one or more events imply that some other event or events have to occur, but not the other way round. For example, a web server that logs an incoming connection should imply that corresponding log lines on the firewall have occurred earlier.
- **Inter-arrival time anomalies** are caused by deviating time intervals between occurrences of log events. They are related to correlation anomalies and may provide additional detection capabilities, e.g., an implied event is expected to occur within a certain time window.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Sequence anomalies** are caused by missing or additional log events as well as deviating orders in sequences of log events that are expected to occur in certain patterns.

Outliers are based on single log line occurrences and are thus the only type of anomalies detectable by static cluster algorithms. All other types of anomalies require dynamic clustering techniques. In addition, anomalies do not necessarily have to be detected using strict rules that report every single violation. For example, event correlations that are expected to occur only in 90% of all cases may be analyzed with appropriate statistical tests.

5.2 Survey Method

In this section we describe our approach to gather and analyze the existing literature.

5.2.1 Set of Criteria

In order to carry out the literature survey on log clustering approaches in a structured way, we initially created a set of evaluation criteria that addresses relevant aspects of the research questions in more detail. The first block of questions in the set of criteria covers purpose, applicability, and usability of the proposed solutions:

P-1 What is the purpose of the introduced approach?

P-2 Does the method have a broad applicability or are there constraints, such as requirements for specific logging standards?

P-3 Is the algorithm a commercial product or has been deployed in industry?

P-4 Is the code of the algorithm publicly accessible?

The next group of questions focuses on the properties of the introduced clustering algorithms:

C-1a What type of technique is applied for static clustering?

C-1b What type of technique is applied for dynamic clustering?

C-2 Is the algorithm fully unsupervised as opposed to algorithms requiring detailed knowledge about the log structures or labeled log data for training?

C-3 Is the clustering character-based?

C-4 Is the clustering word- or token-based?

C-5 Are log signatures or templates generated?

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

C-6 Does the clustering algorithm take dynamic features of log lines (e.g., sequences) into account?

C-7 Does the algorithm generate new clusters online, i.e., in a streaming manner, as opposed to approaches that allocate log lines to a fixed set of clusters generated in a training phase?

C-8 Is the clustering adaptive to system changes, i.e., are existing clusters adjusted over time rather than static constructs?

C-9 Is the algorithm designed for fast data processing?

C-10 Is the algorithm designed for parallel execution?

C-11 Is the algorithm deterministic?

Since we were aware that a large number of approaches aim at anomaly detection, we dedicated the following set of questions to this topic:

AD-1 Is the approach designed for the detection of outliers, i.e., static anomalies?

AD-2 Is the approach designed for the detection of dynamic anomalies?

AD-3 Is the approach designed for the detection of cyber attacks?

Finally, we defined questions that assess whether and how the approaches were evaluated in the respective articles:

E-1 Did the evaluation include quantitative measures, e.g., accuracy or true positive rates?

E-2 Did the evaluation involve qualitative reviews, e.g., expert reviews or discussions of cluster quality?

E-3 Was the algorithm evaluated regarding its time complexity, i.e., running time and scalability?

E-4 Was at least one existing algorithm used as a benchmark for validating the introduced approach?

E-5 Was real log data used as opposed to synthetically generated log data?

E-6 Is the log data used for evaluation publicly available?

The set of evaluation criteria was then completed for every relevant approach. The process of retrieving these articles is outlined in the following section.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.2.2 Literature Search

The search for relevant literature was carried out in November 2019. For this, three research databases were used: (i) ACM Digital Library [ACM], a digital library containing more than 500,000 full-text articles on computing and information technology, (ii) IEEE Xplore Digital Library [IEE], a platform that enables the discovery of scientific articles within more than 4.5 million documents published in the fields computer science, electrical engineering and electronics, and (iii) Google Scholar [Goo], a web search engine for all kinds of academic literature.

The keywords used for searching on these platforms were "log clustering" (29,383 results on ACM, 2,210 on IEEE, 3,050,000 on Google), "log event mining" (54,833 results on ACM, 621 on IEEE, 1,240,000 on Google), "log data anomaly detection" (207,821 results on ACM, 377 on IEEE, 359,000 on Google). We did not make any restrictions regarding the date of publication. The titles and abstracts of the first 300 articles retrieved for each query were examined and potentially relevant documents were stored for thorough inspection. It should be noted that a rather large amount of false positives were retrieved and immediately dismissed. The reason why such unrelated articles appeared is that the keywords in the queries were sometimes misinterpreted by the engines, e.g., results related to "logarithm" showed up when searching for "log". After removing duplicates, this search yielded 207 potentially relevant articles.

During closer inspection, several of these articles were discarded. The majority of these dismissed approaches focused on clustering numeric features extracted from highly structured network traffic logs rather than clustering the raw string messages themselves. This is a broad field of research and there exist numerous papers that apply well-known machine learning techniques for analyzing, grouping, and classifying the parsed data [PES01]. Many other approaches are directed towards process mining from event logs [VdAWM04], which is an extensive topic considered out of scope for our survey since it relies on log traces rather than simple log data. Furthermore, we discarded papers that introduce approaches for analysis and information extraction from log data, but are not fitted for clustering log lines, such as terminology extraction [SBL+09] and compression [BS06]. We also dismissed approaches for clustering search engine query logs [BB00] since they are designed to process keywords written by users rather than log lines generated by programs as outlined in Section 5.1.1. Articles on protocol reverse engineering are discarded, because they are not primarily designed for processing system log lines and surveys on this topic already exist, e.g., [NSC16]. Finally, we excluded articles that do not propose a new clustering approach, but apply existing algorithms without modifications on different data or perform comparisons (e.g., [MZHM+09b]) as well as surveys. This also includes articles that propose algorithms for subsequent analyses such as anomaly detection, alert aggregation, or process mining, that operate on already clustered log data, but do not apply any log clustering techniques themselves.

After this stage, 50 articles remained. A snowball search was conducted with these articles, i.e., articles referenced in the relevant papers as well as articles referencing these

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

papers were individually retrieved. These articles were examined analogously and added if they were considered relevant. Eventually, we obtained 59 articles and 2 tools that were analyzed with respect to the aforementioned characteristics stated in the set of evaluation criteria. We used these criteria to group articles with respect to different features and discover interesting patterns. The following section discusses the findings.

5.3 Survey Results

We arranged the articles into groups according to the properties ascertained in the set of evaluation criteria. We thereby derived common features that could be found in several articles as well as interesting concepts and ideas that stood out from the overall strategies. In the following, we discuss these insights for every group of questions.

5.3.1 Purpose and Applicability (P)

Four main categories of overall design goals (P-1) were identified during the review process:

- *Overview & Filtering.* Log data is usually high-volume data that is tedious to search and analyze manually. Therefore, it is reasonable to reduce the total number of log messages presented to system administrators by removing log events that are frequently repeating without contributing new or any other valuable information. Clustering is able to provide such compact representations of complex log files by filtering out most logs that belong to certain (large) clusters, thus only leaving logs that occur rarely or do not fit into any clusters to be shown to administrators [JSC+09, RJW11].
- *Parsing & Signature Extraction.* These approaches aim at the automatic generation of log event templates (cf. Sect. 5.1.1) for parsing log lines. Parsers enable the allocation of log lines to particular system events, i.e., log line classification, and the structured extraction of parameters. These are important features for subsequent analyses, such as clustering of event sequences or anomaly detection [HZZL17, WLSK19].
- *Outlier Detection.* System failures, cyber attacks, or other adverse system behavior generates log lines that differ from log lines representing normal behavior regarding their syntax or parameter values. It is therefore reasonable to disclose single log lines that do not fit into the overall picture of the log file. During clustering, these log lines are identified as lines that have a high dissimilarity to all existing clusters or do not match any signatures [JSH15, WSL+17].
- *Sequences & Dynamic Anomaly Detection.* Not all adverse system behavior manifests itself as individual anomalous log lines, but rather as dynamic or sequence anomalies (cf. Sect. 5.1.4). Thus, approaches that group sequences of log lines

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

or disclose temporal patterns such as frequent co-occurrence or correlations are required. Dynamic clustering usually relies on line-based event classification as an initial step and often has to deal with interleaving processes that cause interrupted sequences [JYC+17, ABCM09].

Table 5.1 shows the determined classes for each reviewed approach. Note that this classification is not mutually exclusive, i.e., an approach may pursue multiple goals at the same time. For example, [HZZL17] introduce an approach for the extraction of log signatures and then perform anomaly detection on the retrieved events.

As expected, many approaches aim at broad applicability and do not make any specific assumptions on the input data (P-2). Although some authors particularly design and evaluate their approaches in the context of a specific type of log protocol (e.g., router syslogs [QGP+10]), their proposed algorithms are also suitable for any other logging standard. Only few approaches require artifacts specific to some protocol (e.g., Modbus [WLKH18]) for similarity computation or prevent general applicability by relying on labeled data [RJW11] or category labels (e.g., start, stop, dependency, create, connection, report, request, configuration, and other [LLMP05]) for model training, log level information [DC15] for an improved log similarity computation during clustering, or process IDs for linking events to sequences [LZL+16]. Other approaches impose constraints such as the requirement of manually defined parsers [TL10] or access to binary/source code of the log generating system in order to parse logs using the respective print statements [ZZH17, XHF+09, SAvD19].

We mentioned in Section 5.2 that we included two approaches from non-academic literature: Splunk [Car12] and Sequence [Zhe14]. Splunk is a commercial product (P-3) that offers features that exceed log clustering and is deployed in numerous organizations. However, also the authors of scientific papers share success stories about real-world application in their works, e.g., Lin et al. [LZL+16] describe feedback and results following the implementation of their approach in a large-scale environment and Li et al. [LJZ+17] evaluate their approach within a case-study carried out in cooperation with an international company. We appreciate information about such deployments in real-world scenarios, because they validate that the algorithms are meeting the requirements for practical application. Finally, we could only find the original source code of [Zhe14, Vaa03, Vaa04, VP15, MZHM09a, ZX16, TMP17, Shi16, MPB+18, HZZL17, HZH+17, XHF+09] online (P-4). In addition, several reimplementations of algorithms provided by other authors exist. We encourage authors to make their code available open-source in order to enable reproducibility.

5.3.2 Clustering Techniques (C)

In the following, we explore different types of applied clustering techniques with respect to their purpose, their applicability in live systems, and non-functional requirements.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 5.1: Overview of main goals of reviewed approaches (categorizations are not mutually exclusive)

Purpose of approach	**Approaches**
Overview & Filtering	Aharon et al. [ABCM09], Aussel et al. [APC18], Christensen and Li [CL13], Jiang et al. [JHHF08], Joshi et al. [JBG14], Li et al. [LLMP05, LJZ+17], Reidemeister et al. [RJW11], Gainaru et al. [GCTMK11], Gurumdimma et al. [GJL+15], Hamooni et al. [HDX+16], Jain et al. [JSC+09], Jayathilake et al. [JWH17], Leichtnam et al. [LTPM17], Makanju et al. [MZHM09a], Nandi et al. [NMA+16], Ning et al. [NJCY14], Qiu et al. [QGP+10], Ren [RCY+18], Salfner and Tschirpke [ST08], Schipper et al. [SAvD19], Splunk [Car12], Taerat et al. [TBG+11], Xu et al. [XHF+09], Zou et al. [ZQJ16]
Parsing & Signature Extraction	Agrawal et al. [AKG19], Chuah et al. [CKH+10], Du and Li [DL16], Fu et al. [FLWL09], Gainaru et al. [GCTMK11], Hamooni et al. [HDX+16], He et al. [HZZL17, HZH+17], Jayathilake et al. [JWH17], Kimura et al. [KIM+14], Kobayashi et al. [KFE14], Li et al. [LJZ+17], Li et al. [LDF+18], Makanju et al. [MZHM09a], Messaoudi et al. [MPB+18], Menkovski and Petkovic [MP17], Mizutani [Miz13], Nagappan and Vouk [NV10], Nandi et al. [NMA+16], Ning et al. [NJCY14], Qiu et al. [QGP+10], Sequence [Zhe14], Shima [Shi16], Taerat et al. [TBG+11], Tang and Li [TL10], Tang et al. [TLP11], Thaler et al. [TMP17], Tovarňák et al. [TP19], Vaarandi [Vaa03, Vaa04], Vaarandi and Pihelgas [VP15], Wurzenberger et al. [WLSK19], Zhang et al. [ZZH17], Zhao and Xiao [ZX16], Zulkernine et al. [ZMP+13]
Outlier Detection	Juvonen et al. [JSH15], Leichtnam et al. [LTPM17], Splunk [Car12], Wurzenberger et al. [WSFK17, WSL+17]
Sequences & Dynamic Anomaly Detection	Aharon et al. [ABCM09], Chuah et al. [CKH+10], Du et al. [DLZS17], Du and Cao [DC15], Fu et al. [FLWL09], Gurumdimma et al. [GJL+15], He et al. [HZZL17, HZH+17], Jia et al. [JYC+17], Kimura et al. [KIM+14], Li et al. [LDF+18], Lin et al. [LZL+16], Nandi et al. [NMA+16], Salfner and Tschirpke [ST08], Splunk [Car12], Stearley [Ste04], Vaarandi [Vaa04], Wang et al. [WLKH18], Xu et al. [XHF+09], Zhang et al. [ZZH17], Zhang et al. [ZXL+19], Zou et al. [ZQJ16]

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Types of Static Clustering Techniques

One of the most interesting findings of this research study turned out to be the large diversity of proposed clustering techniques (C-1a, C-1b). Considering static clustering approaches, a majority of the approaches employ a distance metric that determines the similarity or dissimilarity of two or more strings. Based on the resulting scores, similar log lines are placed in the same clusters, while dissimilar lines end up in different clusters. The calculation of the distance metric may thereby be character-based, token-based or a combination of both strategies (C-3, C-4). While token-based approaches assume that the log lines can reasonably be split by a set of predefined delimiters (most frequently, only white space is used as a delimiter), character-based approaches are typically more flexible, but also computationally more expensive. For example, Juvonen et al. [JSH15] and Christensen and Li [CL13] compute the amount of common n-grams between two lines in order to determine their similarity. Du and Cao [DC15], Ren et al. [RCY+18], Salfner and Tschirpke [ST08], and Wurzenberger et al. [WSFK17, WSL+17] use the Levenshtein metric to compute the similarity between two lines by counting the character insertions, deletions and replacements needed to transform one string into the other. Taerat et al. [TBG+11], Gurumdimma et al. [GJL+15], Jain et al. [JSC+09], Zou et al. [ZQJ16], and Fu et al. [FLWL09] employ a similar metric based on the words of a line rather than its characters. Another simple token-based approach for computing the similarity between two log lines is by summing up the amount of matching words at each position. In mathematical terms, this similarity between log lines a and b with their respective tokens $a_1, a_2, ..., a_n$ and $b_1, b_2, ..., b_m$ is computed by $\sum_{i=1}^{min(n,m)} \mathbb{I}(a_i, b_i)$, where $\mathbb{I}(a_i, b_i)$ is 1 if a_i is equal to b_i and 0 otherwise. This metric is frequently normalized [NJCY14, ABCM09, HZZL17, LDF+18, Miz13] and weighted [TL10, HDX+16]. Joshi et al. [JBG14] use bit patterns of tokens to achieve a similar result. Li et al. [LJZ+17] compute the similarity between log lines after transforming them into a tree-like structure. Du and Cao [DC15] also consider the log level (e.g., INFO, WARN, ERROR) relevant for clustering and point out that log lines generated on a different level should not be grouped together. Finally, token vectors that emphasize the occurrence counts of words rather than their positions (i.e., the well-known bag of words model) may be used to compute the cosine similarity [LZL+16, Shi16, Car12] or apply k-means clustering [APC18].

Not all approaches employ distance or similarity metrics. SLCT [Vaa03] is one of the earliest published approaches for log clustering. The idea behind the concept of SLCT is that frequent tokens (i.e., tokens that occur more often than a user-defined threshold) represent fixed elements of log templates, while infrequent tokens represent variables. Despite being highly efficient, one of the downsides of SLCT is that clustering requires three passes over the data: The first pass over all log lines retrieves the frequent tokens, the second pass generates cluster templates by identifying these frequent tokens in each line and filling the gaps with wildcards, and the third pass reports cluster templates that represent sufficiently many log lines. Allocating the log lines to clusters is accomplished during the second pass, where each log line is assigned to the an already existing or newly generated template.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Table 5.2: Assessed properties regarding clustering techniques assigned to each approach

Approach	C-1a	C-1b	C-2	C-3	C-4	C-5	C-6	C-7	C-8	C-9	C-10
[AKG19]	5, 11		✓		✓	✓		✓	✓	✓	✓
[ABCM09]	1	1	✓		✓	✓	✓	✓		✓	
[APC18]	2, 11		✓		✓					~	
[CL13]	1		✓	✓				✓	✓	~	✓
[CKH+10]	11	2, 9	~		✓	✓	✓	✓		~	
[DL16]	1, 5		✓		✓	✓		✓	✓	✓	
[DLZS17]	[DL16]	2, 3, 9	✓		✓		✓	✓	~	~	
[DC15]	1, 11	1, 2	✓	✓			✓				
[FLWL09]	1, 11	9	✓		✓	✓	✓		✓		✓
[GCTMK11]	4		✓		✓	✓		✓	✓	~	
[GJL+15]	1	1, 2, 9	✓		✓		✓				
[HDX+16]	1		✓		✓	✓		✓	✓	✓	✓
[HZH+17]	4, 5		✓		✓	✓				✓	✓
[HZZL17]	1		✓		✓	✓		✓	✓	✓	
[JSC+09]	1	1, 2	✓		✓		✓	✓	✓	~	
[JWH17]	5		✓	✓	✓	✓					
[JYC+17]	[Vaa03]	2, 9	✓	✓	✓	✓	✓			✓	✓
[JHHF08]	11		✓		✓	✓				✓	
[JBG14]	1		✓		✓	✓		✓	✓	✓	
[JSH15]	1		✓	✓				✓			
[KIM+14]	9	9	✓		✓	✓	✓			~	
[KFE14]	3				✓	✓					
[LTPM17]	10				✓						
[LLMP05]		9			✓		✓			~	
[LJZ+17]	1	9	✓		✓	✓	✓			~	
[LDF+18]	1, 11	3, 9	✓		✓	✓	✓	✓	✓	~	
[LZL+16]	[FLWL09]	1	✓		✓		✓				
[MZHM09a]	4		✓		✓	✓				✓	
[MP17]	1, 3		✓		✓	✓				~	
[MPB+18]	7		✓		✓	✓				~	
[Miz13]	1		✓	✓	✓	✓		✓	✓	✓	
[NV10]	2		✓		✓	✓				✓	
[NMA+16]	1, 2, 5	9	✓	✓	✓	✓	✓	✓		✓	✓
[NJCY14]	1		✓		✓	✓		✓	✓		✓
[QGP+10]	2, 11	9	✓		✓	✓	✓			~	
[RJW11]	1, 2, 5			✓	✓	✓				✓	
[RCY+18]	1, 3, 11			✓	✓					~	
[ST08]	1	9	✓	✓	✓		✓			~	
[SAvD19]	6		✓	✓	✓	✓				~	

Continued on next page

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Table 5.2 – continued from previous page

Approach	C-1a	C-1b	C-2	C-3	C-4	C-5	C-6	C-7	C-8	C-9	C-10
[Zhe14]	11		✓	~	✓	✓		✓	✓	✓	
[Shi16]	1		✓		✓	✓		✓	✓	~	
[Car12]	1		✓		✓			✓		✓	
[Ste04]	2, [Vaa03]	9	✓		✓	✓	✓				
[TBG+11]	1		✓		✓	✓		✓		~	
[TL10]	1		~		✓	✓		✓		✓	
[TLP11]	5		✓		✓	✓				✓	
[TMP17]	3			✓		✓				~	
[TP19]	2, [NV10]		✓		✓	✓				✓	✓
[Vaa03]	2		✓		✓	✓				✓	
[Vaa04]	[Vaa03]	8	✓		✓	✓	✓			✓	
[VP15]	2		✓		✓	✓				✓	
[WLKH18]	1	5	✓		✓	✓	✓			~	
[WSFK17]	1		✓	✓						✓	
[WSL+17]	1		✓	✓				✓		✓	
[WLSK19]	2		✓		✓	✓				~	
[XHF+09]	6	2, 9	✓		✓	✓	✓			✓	✓
[ZZH17]	6	6	✓		✓	✓	✓			✓	
[ZXL+19]	2, 11, [HZZL17]	3	✓		✓	✓	✓	✓	✓	~	
[ZX16]	2, 11		✓		✓	✓					
[ZQJ16]	1	9	~	✓	✓	✓	✓		~	~	
[ZMP+13]	2	9	✓	✓	✓	✓	✓	✓		✓	

Density-based clustering appears to be a natural strategy for generating trees [QGP+10, ZX16, WLSK19, TP19], i.e., data structures that represent the syntax of log data as sequences of nodes that branch into subsequences to describe different log events. Thereby, nodes represent fixed or variable tokens and may even differentiate between data types, e.g., numeric values or IP addresses. The reason why all of the reviewed approaches leveraging trees use density-based techniques is likely attributable to the way trees are built: Log messages are processed token-wise from their beginning to their end; identical tokens in all lines are frequent tokens that result in fixed nodes, tokens with highly diverse values are infrequent and result in variable nodes, and cases in between result in branches of the tree.

Types of Dynamic Clustering Techniques

Several approaches pursue the clustering of log sequences rather than only grouping single log lines (C-6). Thereby, process IDs that uniquely identify related log lines may be exploited to retrieve the sequences [LZL+16]. For example, Fu et al. [FLWL09] use these IDs to build a finite state automaton describing the execution behavior of the monitored system. However, logs that do not contain such process IDs require mechanisms for detecting relations between identified log events. Du and Cao [DC15] and

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

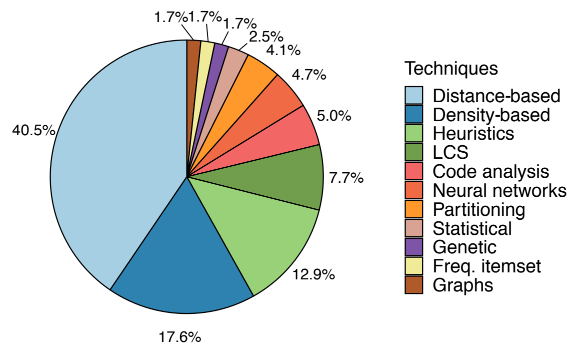


Figure 5.4: Relative frequencies of static clustering techniques used in the reviewed articles.

Gurumdimma et al. [GJL+15] first cluster similar log lines, then generate sequences by grouping events occurring in time windows and finally cluster the identified sequences in order to derive behavior patterns. Similarly, Salfner and Tschirpke [ST08] group generated events that occur within a predefined inter-arrival time and cluster the sequences with a hidden semi-Markov Model. Also Qiu et al. [QGP+10] measure the inter-arrival time of log lines for clustering periodically occurring events and additionally group the events by derived correlation rules. Kimura et al. [KIM+14] derive event co-occurrences by factorizing a 3-dimensional tensor consisting of the previously identified templates, hosts and time windows. DeepLog [DLZS17] extends Spell [DL16] by computing probabilities for transitions between the identified log events in order to construct a workflow model. Jain et al. [JSC+09] group time-series derived from cluster appearances in a hierarchical fashion. LogSed [JYC+17] and OASIS [NMA+16] analyze frequent successors and predecessors of lines for mining a control flow graph. After first categorizing log messages using probabilistic models [LLMP05] and distance-based strategies [LJZ+17], the authors determine the temporal relationships between log events by learning the distributions of their lag intervals, i.e., time periods between events. Other than the previous approaches, Aharon et al. [ABCM09] assume that the order of log lines is meaningless and their algorithm PARIS thus identifies log events that frequently occur together within certain time windows regardless of their order.

We summarize the results in Table 5.2. For columns C-1a and C-1b, we coded distance-based strategies as (1) and density-based strategies as (2). Note that for static clustering, distances are usually measured between log lines and densities refer to token frequencies, while for dynamic clustering techniques, distances are computed between time-series of event occurrences and densities refer to event frequency counts. Other identified strategies used for static and dynamic clustering are (3) Neural Networks, which are useful for signature extraction [KFE14, MP17, TMP17] and event classification [RCY+18] by Natural Language Processing (NLP) as well as for detecting sequences in the form of Long Short-Term Memory (LSTM) recurrent neural networks [DLZS17, LDF+18, ZXL+19], (4) iterative partitioning, where groups of log lines are recursively split into subgroups according to particular token positions [MZHM09a, GCTMK11, HZH+17], (5) Longest

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Common Substring (LCS), which is a measure for the similarity of log lines [AKG19, DL16, HZH+17, JWH17, RJW11, TLP11] or sequences of log events [WLKH18], (6) binary/source code analysis [XHF+09, ZZH17, SAvD19], (7) genetic algorithms [MPB+18], (8) frequent itemset mining [Vaa04], (9) statistical modeling [DLZS17, KIM+14, LLMP05, LJZ+17, LDF+18], and (10) graph community extraction [LTPM17]. In addition, a number of approaches employ (11) heuristics for replacing tokens with wildcards if they match specific patterns, e.g., IP addresses or numeric values that most likely represent IDs. While such rules are frequently only used for preprocessing log data before clustering, the approaches by Chuah et al. [CKH+10] and Jiang et al. [JHHF08] suggest that heuristics alone may be sufficient to generate templates. Figure 5.4 shows a visual overview of the techniques used in static log clustering. The plot shows that distance-based and density-based techniques are the most common techniques, being used in more than half of all reviewed approaches. Dynamic clustering techniques are less diverse: Most approaches apply statistical methods to generate links between events and rely on event count matrices for grouping and anomaly detection.

Applicability in Live Systems

Almost all approaches employ self-learning techniques that operate in an unsupervised fashion, i.e., no labeled training data is required for building the model of normal system behavior (C-2). This corresponds to the mentioned ambition of proposing algorithms that are mostly independent of the log structure and allow automatic processing with minimal human interference. However, we identified some approaches that do not follow this tendency and need labeled data for training: Kobayashi et al. [KFE14] use templates that define which tokens in log messages are fixed or variable, Thaler et al. [TMP17] also use such templates but mark every character of the log message as fixed or variable, Li et al. [LLMP05] use categorical states that describe the type of log line, and Reidemeister et al. [RJW11] use labels that describe types of failures. Other approaches rely on extensive manual work preceding clustering, including the manual extraction of relevant attributes into a common format [LTPM17] or the definition of parsers [TL10]. Similarly, Chuah et al. [CKH+10] and Zou et al. [ZQJ16] incorporate domain knowledge of the log structure in the clustering procedure.

Most articles lack precise investigations of running time complexities and upper bounds due to algorithmic complexity and parametric dependencies. However, we observed that some of the proposed approaches are particularly designed for online clustering (C-7), while others pursue offline or batch clustering. Online clustering means that at any given point in time during the clustering, all the processed log lines are already allocated to clusters. This usually implies that the running time grows at most linearly with the number of processed lines, which is an important property for many real-world applications where log lines are processed in streams rather than limited sets. Note that an allocation of lines to existing clusters in a streaming manner is almost always possible and we therefore only considered approaches that are able to generate new clusters on the fly as capable of online-processing. Typically, the reviewed online algorithms

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

proceed in the following way: First, an empty set of clusters is initialized. Then, for each newly processed log line, the algorithm attempts to find a fitting cluster in the set of clusters. If such a cluster is found, the log line is allocated to it; otherwise, a new cluster containing that line is created and added to the set of clusters. This step is repeated indefinitely [ABCM09].

In addition to generating new clusters, approaches that we consider adaptive are also able to modify existing cluster templates when new log lines are received (C-8). Such adaptive approaches are in particular useful when being employed in systems that undergo frequent changes, e.g., software upgrades or source code modifications that affect the logging behavior [ZXL+19, GCTMK11]. While non-adaptive approaches usually require a complete reformation of all clusters and templates, adaptive approaches dynamically adjust to the new baseline without the need of instantly "forgetting" all previously learned patterns. Approaches that do not aim at the generation of log templates may achieve adaptive behavior by only considering the most recently added log lines as relevant for clustering [CL13].

Non-functional Requirements

The further columns provide information on whether the approaches were particularly designed for high efficiency (C-9) or parallel execution (C-10). Note that we considered a comparative evaluation on the efficiency of all algorithms out of scope for this survey, but rather assessed whether the authors particularly designed the algorithm for high log throughput, for example, by employing data structures or methods that enable fast data processing. In general, such an evaluation is difficult, because the running time often depends on the type of log data, parameter settings and data preprocessing.

Finally, we assessed that most algorithms operate in a deterministic fashion (C-11). However, some exceptions particularly make use of randomization, for example genetic algorithms [MPB+18], randomized hash functions [JBG14], randomly initialized fields [JSH15] and all approaches that rely on neural networks.

5.3.3 Anomaly Detection (AD)

According to our set of evaluation criteria, we group the approaches with respect to their ability to detect static or dynamic anomalies and discuss the origin of anomalies that are typically detected in the reviewed articles.

Static Outlier Detection

As mentioned before, not all reviewed articles primarily pursue anomaly detection (cf. Table 5.1) and thus do not include discussions about the effectiveness of their detection capabilities. However, the patterns or groups of log lines resulting from the clustering can always be used for the detection of anomalies. For example, log lines that are very dissimilar to all clusters or do not match any of the retrieved patterns are considered

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

outliers (AD-1). New and previously unseen lines are usually regarded as suspicious and should be reported. In addition, clusters that are unusually small or very distant to all other clusters may indicate anomalous groups of log lines. Clearly, domain knowledge is required to interpret the retrieved lines and Hamooni et al. [HDX+16] add that a keyword search on the new logs is an effective measure for system administrators to locate and interpret the occurred event.

SLCT [Vaa03] and LogCluster [VP15] allocate all log lines in an outlier cluster if they do not match any of the generated log templates, i.e., patterns that represent each cluster. They used logs collected from a mail server and found that the identified outliers correspond to errors and unauthorized access attempts. In a similar manner, Wurzenberger et al. [WSL+17], Stearley [Ste04] and Splunk [Car12] identify rare log lines that do not end up in large clusters as outliers. HLAer [NJCY14] offers two possibilities for outlier detection: an online method based on pattern matching as well as an offline method that uses the same similarity score used for clustering. Similarly, Wurzenberger et al. [WSFK17] defines a similarity function for outlier detection and further mentions that small clusters contain potentially interesting log lines. Juvonen et al. [JSH15] detect outliers without the need for pattern matching. They inject cross-site scripting (XSS) attacks and the resulting log lines are located far away from all the other log lines when being projected into an euclidean space.

Dynamic Anomaly Detection

Other than detecting outliers, some algorithms aim at the detection of failure patterns (AD-2). Thereby, the retrieval of distinct and expressive descriptors is regarded as the main goal. For example, Baler [TBG+11] identifies patterns corresponding to failure modes of the system CPU and memory errors. Such fault types are also detected by Zou et al. [ZQJ16] who group alerts that occur within time windows. Categories of these alerts thereby include errors caused by the network, failed authentications, peripheral devices and the web engine.

In addition, some approaches support root-cause analysis, where the identification of log events occurring in the past that relate to detected failures is pursued. Thereby, algorithms utilize the learned temporal dependencies between log events for such reasoning. Chuah et al. [CKH+10] and Kimura et al. [KIM+14] particularly focus on root-cause analysis and identify temporal dependencies by correlating event occurrences within time windows. However, Li et al. [LLMP05, LJZ+17] point out that a correct selection of the time window sizes is often difficult, and therefore propose a solution that relies on lag time distributions rather than time windows.

It is non-trivial to derive dynamic properties from clusterings. LogTree [TL10] supports manual detection by displaying patterns of cluster appearances. In their case study, misconfigurations in HTML files were detected. For an analytical detection, Drain [HZZL17] gradually fills an event count matrix that keeps track of the number of occurrences of each log event. They then use principal component analysis for detecting

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

unusual points in the resulting matrix. Similarly, Xu et al. [XHF+09] use PCA for detecting anomalies in high-dimensional message count vectors and additionally consider state variables for filling the matrix. Du and Cao [DC15] detect anomalous system behavior by applying a distance metric on time-series derived from event frequencies.

Beside unusual frequencies of occurring events, the execution order of certain log line types may be used as another indicator for anomalies. Zulkernine et al. [ZMP+13] derive correlation rules from line patterns that frequently occur together. Fu et al. [FLWL09] learn the execution order of events and detect deviations from this model as work flow errors. In addition, they identify performance issues by measuring the execution times of newly occurring log sequences and compare them with the learned behavior. Jia et al. [JYC+17] also detect unknown logs as redundancy failures, deviations from execution orders as sequence anomalies and deviations from interval times as latency anomalies. Beside sequence errors, Nandi et al. [NMA+16] make use of a control flow-graph in order to also detect changes of branching distributions, i.e., changes of occurrence probabilities of certain log events in sequences. DeepLog [DLZS17] trains a Long Short-Term Memory (LSTM) neural network with such workflow transition probabilities and automatically detects any deviations from the learned behavior. A different approach is taken by Gurumdimma et al. [GJL+15], who detect failure patterns of sequences rather than single events.

Cyber Attack Detection

Finally, we noted that although many approaches are directed towards anomaly detection, these anomalies are almost always considered to be random or naturally occurring failures rather than targeted cyber attacks (AD-3), such as denial-of-service and scanning attacks [WLKH18, DLZS17], injections [JSH15, WSL+17], or unauthorized accesses [RJW11, Vaa03]. We assume that the reasons for this trend are manifold: (i) Failures may be more common than attacks in the considered systems and thus pose a higher risk, (ii) attacks are implicitly assumed to produce artifacts similar to failures and can thus be detected using the same methods, and (iii) lack of log files containing cyber attacks for evaluation.

5.3.4 Evaluation (E)

In the following, we investigate the procedures of evaluating approaches presented in the reviewed papers. In particular, we discuss what kind of evaluation techniques were applied to assess fulfillment of functional and non-functional requirements, and whether the results are reproducible.

Evaluation Techniques

Every reviewed clustering approach includes at least some kind of experiments and discussion of results. As shown in Table 5.3, a majority of authors use quantitative metrics for validating and evaluating their proposed concepts (E-1). We identified three main approaches to quantitative evaluation: (i) Unsupervised methods that do not

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

require a labeled ground truth for comparison. There exist various possibilities for estimating the quality of the clustering in an unsupervised fashion. Menkovski and Petkovic [MP17] show that the consistency of the clustering can be assessed by measures such as the Silhouette Coefficient [Rou87], which measures the relation between inter- and intra-cluster distances. Kimura et al. [KIM+14] estimate the quality of the log templates heuristically, by assuming that all tokens without numbers should end up as fixed elements in the generated templates. This measure is easy to compute, but has the disadvantages that it may produce incorrect results in cases where the heuristics do not apply and that it is only reasonably applicable for log files where such heuristics are known to fit the data. Alternatively, Li et al. [LJZ+17] make use of event coverage during clustering, a measure for the goodness of a set of cluster descriptors with respect to their similarities to all log lines. The problem with such strategies is that it is typically difficult to obtain interpretable and comparable results and thus most of the reviewed approaches do not take unsupervised evaluation into consideration. (ii) The grouped log lines are compared to a manually crafted ground truth of cluster allocations. This allows the computation of the accuracy, precision, recall and F-score of the approach. Different strategies for computing these metrics are possible. For example, He et al. [HZZL17] count two log lines generated by the same event grouped in the same cluster as true positive; two lines generated by different events grouped in the same cluster as false positive and two lines generated by the same event grouped in different clusters as false negative. Contrary to such a line-based measure, Du and Li [DL16] evaluate their approach with a more strict focus on clusters. They measure the accuracy by the number of lines allocated to correct clusters, where a cluster is counted correct if all and only all log lines of a particular type from the ground truth are allocated to the same cluster. The results of line-based and cluster-based evaluations can be very different: Consider a clustering result containing one large cluster. A line-based accuracy measure will show good results as long as many log lines of that type end up in the same clusters, even if a portion of the lines end up in other clusters or a few misclassifications occurred. The accuracy measured in cluster-based evaluation on the other hand will indicate poor results when only one or few misclassifications occur in that cluster, since this causes that all contained lines are considered as incorrectly classified. Kobayashi et al. [KFE14] measure the accuracy by inspecting the templates rather than the associated log lines. In particular, they count the number of fields correctly identified as fixed or variable in each generated log template. Hamooni et al. [HDX+16] apply a similar approach but also take types of fields, e.g., string, number or IP, into account. This approach appears particularly useful when obtaining a ground truth or labeling all log lines is not possible, but the number and structure of expected cluster templates can be determined. (iii) The quality of the clustering is assessed by its ability to detect anomalies. In this case, a ground truth of known anomalies is required for counting the true positives (correctly identified anomalies), false positives (incorrectly identified anomalies), false negatives (missed anomalies), and true negatives (correctly classified non-anomalous instances). The advantage of this method is that it does not require labels for log lines or knowledge about all clusters. However, anomaly-based evaluation relies on a dataset containing

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

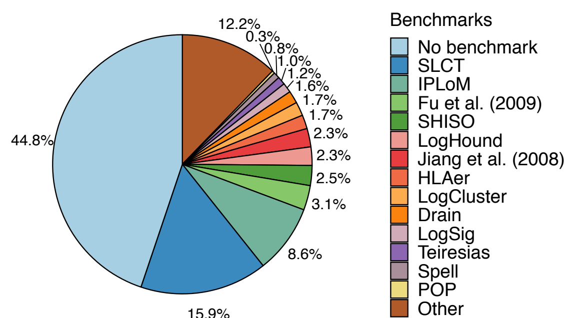


Figure 5.5: Relative frequencies of benchmarks used for evaluation.

anomalies and only measures the quality of the clustering indirectly, i.e., it is possible that an inappropriate detection mechanism is responsible for a poor detection accuracy even though the clustering is of good quality.

A number of approaches also qualitatively assess the clustering (E-2). This is especially common for approaches that aim at the extraction of log signatures. For example, Taerat et al. [TBG+11] discuss the appropriateness of the number of clusters and outliers based on domain knowledge about the used log data. Moreover, they manually check whether unnecessary signatures exist or generated patterns are too general and thus lead to overgrouped clusters. In cases where a ground truth of expected signatures is available, differences and overlaps between the generated and expected patterns can be determined (e.g., Fu et al. [FLWL09]). Because of the ambiguities of what is considered an appropriate clustering, experts or administrators with domain knowledge about the specific real-world use cases are occasionally consulted for labeling the data [XHF+09] or validating the results [LZL+16, ABCM09, LLMP05, LJZ+17, QGP+10, Ste04].

Evaluation of Non-functional Requirements

Given that many approaches are particularly designed for fast processing of log lines, a high number of articles also include an empirical evaluation of running time requirements (E-3). Thereby, both the total time necessary to process a specific log file [MPB+18] as well as the scalability of the algorithm with respect to the number of processed log lines [Miz13] are relevant characteristics.

Comparisons and Reproducibility

Most evaluations include thorough comparisons with one or multiple widely-applied approaches (E-4). For example, HLAer [NJCY14] is compared by [HDX+16] with their algorithm LogMine regarding the accuracy of the generated signatures and SLCT [Vaa03] is used as a benchmark by Joshi et al. [JBG14] for comparing the quality of the clustering and by Stearley [Ste04] regarding outlier detection. Figure 5.5 shows an overview of approaches that are frequently used as benchmarks. Note that it is common that more than one approach is used for comparison, in which case the approaches were

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 5.3: Assessed properties regarding the evaluation carried out in each approach

Approach	**E-1**	**E-2**	**E-3**	**E-4**	**E-5**	**E-6**
Agrawal et al. (Logan) [AKG19]	✓	✓	✓	✓	✓	
Aharon et al. (PARIS) [ABCM09]		✓		✓	✓	
Aussel et al. [APC18]	✓			✓	✓	
Christensen and Li [CL13]	✓		∼	✓	✓	
Chuah et al. (Fdiag) [CKH+10]		✓			✓	✓
Du and Li (Spell) [DL16]	✓		✓	✓	✓	✓
Du et al. (DeepLog) [DLZS17]	✓	✓				✓
Du and Cao [DC15]	✓			✓	✓	✓
Fu et al. [FLWL09]	✓	✓		✓		
Gainaru et al. (HELO) [GCTMK11]	✓			✓	✓	✓
Gurumdimma et al. [GJL+15]	✓			✓	✓	✓
Hamooni et al. (LogMine) [HDX+16]	✓		✓	✓	✓	
He et al. (POP) [HZZL17]	✓		✓	✓	✓	✓
He et al. (Drain) [HZH+17]	✓		✓	✓	✓	✓
Jain et al. [JSC+09]	✓		✓		✓	✓
Jayathilake et al. [JWH17]					✓	
Jia et al. (LogSed) [JYC+17]	✓		✓		✓	✓
Jiang et al. [JHHF08]	✓	✓		✓	✓	✓
Joshi et al. [JBG14]	✓			✓	✓	
Juvonen et al. [JSH15]			✓		✓	
Kimura et al. [KIM+14]	✓	✓			✓	
Kobayashi et al. [KFE14]	✓		✓	✓	✓	
Leichtnam et al. (STARLORD) [LTPM17]		✓			✓	✓
Li et al. [LLMP05]	✓	✓			✓	
Li et al. (FLAP) [LJZ+17]	✓	✓	✓		✓	
Li et al. [LDF+18]		✓	✓		✓	
Lin et al. (LogCluster) [LZL+16]	✓	✓		✓	✓	
Makanju et al. (IPLoM) [MZHM09a]	✓			✓	✓	✓
Menkovski and Petkovic [MP17]	✓		✓	✓	✓	✓
Messaoudi et al. (MoLFI) [MPB+18]	✓			✓	✓	
Mizutani (SHISO) [Miz13]		✓	✓	✓	✓	✓
Nagappan and Vouk [NV10]				✓	✓	
Nandi et al. (OASIS) [NMA+16]	✓		✓		✓	
Ning et al. (HLAer) [NJCY14]		✓	✓	✓	✓	✓
Qiu et al. [QGP+10]	✓	✓			✓	
Reidemeister et al. [RJW11]	✓			✓		
Ren et al. [RCY+18]	✓			✓	✓	✓
Salfner and Tschirpke [ST08]	✓				✓	
Schipper et al. [SAvD19]	✓		✓		✓	

Continued on next page

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 5.3 – continued from previous page

Approach	E-1	E-2	E-3	E-4	E-5	E-6
Sequence [Zhe14]						
Shima (LenMa) [Shi16]		✓	✓	✓	✓	✓
Splunk [Car12]						
Stearley (Teiresias) [Ste04]		✓		✓	✓	
Taerat et al. (Baler) [TBG$^+$11]		✓		✓	✓	
Tang and Li (LogTree) [TL10]	✓		✓	✓	✓	✓
Tang et al. (LogSig) [TLP11]	✓		✓	✓	✓	
Thaler et al. [TMP17]	✓			✓		
Tovarňák et al. [TP19]	✓	✓	✓	✓	✓	✓
Vaarandi (SLCT) [Vaa03]			✓		✓	
Vaarandi (LogHound) [Vaa04]			✓		✓	
Vaarandi and Pihelgas (LogCluster) [VP15]			✓	✓	✓	
Wang et al. [WLKH18]	✓					
Wurzenberger et al. [WSFK17]	✓		✓			
Wurzenberger et al. [WSL$^+$17]	✓					
Wurzenberger et al. (AECID-PG) [WLSK19]	✓			✓	✓	✓
Xu et al. [XHF$^+$09]	✓	✓	✓		✓	
Zhang et al. (GenLog) [ZZH17]	✓	✓	✓		✓	✓
Zhang et al. (LogRobust) [ZXL$^+$19]	✓				✓	✓
Zhao and Xiao [ZX16]	✓		✓		✓	
Zou et al. (UiLog) [ZQJ16]	✓	✓	✓		✓	✓
Zulkernine et al. (CAPRI) [ZMP$^+$13]		✓		✓	✓	✓

added in proportionally. As visible in the plot, the most frequently used algorithms for benchmarking are SLCT [Vaa03] and IPLoM [MZHM09a]. It is also remarkable that all approaches visible in the plot are mainly used for signature extraction. This suggests that there exist more renowned standards for signature extraction than for other clustering approaches. It must however be noted that a majority of the reviewed articles employ signature extraction and thus dominate this statistic.

Most of the articles were evaluated with logs collected from real-world computer systems (E-5). Due to confidentiality of these logs, not all of them are publicly available (E-6). The most common open-source datasets used in the reviewed articles are the supercomputer logs [CFD] Blue Gene/L, Thunderbird, RedStorm, Spirit, Liberty, etc. used by [AKG19, CKH$^+$10, GCTMK11, HZZL17, HZH$^+$17, NJCY14, MZHM09a, MPB$^+$18, JSC$^+$09, GJL$^+$15, DL16, DC15, JHHF08, TP19, WLSK19]; other sources are Hadoop Distributed File System (HDFS) logs [Wei09] used by [AKG19, APC18, DLZS17, HZZL17, MPB$^+$18, XHF$^+$09, ZXL$^+$19, TP19, WLSK19], system logs [Chu10] used by [Miz13, Shi16], and web logs [Capb] used by [ZMP$^+$13]. The artificially generated network logs data [GCH$^+$11] used by [DLZS17] is particularly interesting, because it comes with a ground truth and information on the attacks that were injected during the data collection. Figure 5.6 shows an overview of log data sources used in the reviewed papers. Note that

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

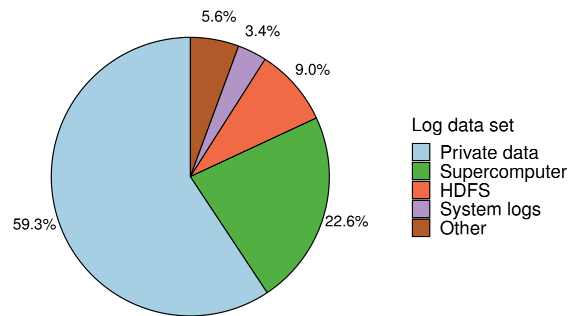


Figure 5.6: Relative frequencies of log data used for evaluation.

approaches that use multiple logs, e.g., supercomputer and HDFS logs, were added in proportionally, and evaluation on non-available data ("Private data") was only counted when there was no evaluation on publicly available data. As visible in the plot, almost 60% of the reviewed approaches involve non-reproducible evaluation.

5.3.5 Discussion

Based on the results outlined in the previous section, several findings can be derived. In the following, we discuss identified issues with the overall problem of clustering log data, disadvantages of employed clustering techniques, and frequently encountered issues in evaluation.

Problem domains

We did not expect to see such a high number of articles primarily focused on the extraction and generation of signatures, while comparatively few articles are oriented towards anomaly detection. Especially static outlier detection, i.e., the identification of log lines with unusual structure or content, appears to be more of a by-product rather than a main feature of signature generating approaches. This may of course be attributable to the fact that such a detection is often a trivial subsequent step to any clustering method. On the other hand, dynamic anomaly detection such as correlation analysis and especially the identification of sequences of log lines appears to be of a higher relevance and the problem of missing sequence identifiers (process IDs discussed in Section 5.1.1) is tackled with various strategies.

Techniques

We were surprised to observe discords regarding some general assumptions on the nature of log files. First of all, we noted a tendency to employ token-based approaches rather than character-based approaches. We attribute this to the fact that token-based strategies are generally computationally less expensive and align better with heuristics, for example, replacing numeric values with wildcards before carrying out a more sophisticated clustering procedure. Despite these advantages, we think that character-based approaches have

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

high potential of generating more precise cluster templates. We have already pointed out in [WSFK17] that token-based approaches are unable to correctly handle words that differ only slightly, e.g., URLs or words with identical semantic meaning such as "u.user" and "t.user" that are frequently found in SQL logs. Moreover, choosing a set of delimiters that are used to split the log messages into tokens is not trivial in practice, because different sets of delimiters may be required for appropriately tokenizing different log messages [JVH+19, TP19].

We also observed that several token-based algorithms compare tokens only at identical positions. The problem with such a strategy is that optional tokens or tokens consisting of multiple words shift the positions of the remaining tokens in the log line. This may cause otherwise similar log lines to incorrectly end up in different clusters [TP19]. While some articles such as Makanju et al. [MZHM09a] explicitly state or implicitly assume that the order of words is relevant for clustering, others (e.g., Vaarandi and Pihelgas [VP15]) particularly design their algorithms to be resilient to word shifts. Since optional words and free text are common in most unstructured log files, we recommend to carry out research on approaches that alleviate these issues.

We stated in Sect. 5.2.2 that approaches on protocol reverse engineering [NSC16] are excluded from this survey, because of their focus on network protocols rather than system log data. However, we see the application of algorithms from protocol reverse engineering for the generation of log signatures as a potentially interesting area for future research. The reason for this is that existing protocol reverse engineering approaches often consider both character-based matching through n-grams as well as the positions of these n-grams or tokens relevant for the extraction of protocol syntaxes. Adapting concepts from protocol reverse engineering may thus effectively alleviate the previously described issues with existing log signature extraction approaches.

Benchmarking & Evaluation

Despite of the fact that SLCT [Vaa03] is one of the first algorithms designed for log clustering, it is still regarded as de facto standard due to its open-source availability. However, more recent articles demonstrated its weaknesses and proposed alternative clustering strategies that largely improved the quality of clusters and signatures. In particular, SLCT generates overly general clusters consisting of only wildcards, which obviously cover a large number of log lines but provide little to no information for the user, as well as overly specific patterns of similar log lines [TBG+11]. We therefore suggest to employ more recent alternative approaches for benchmarking in future articles. In addition, SLCT and other standards such as LogHound [Vaa04], LogCluster [VP15], and IPLoM [MZHM09a] only operate on fixed-size log files and are not able to incrementally process log lines for clustering. However, since stream processing is essential for grouping logs in real-world environments where frequent reclusterings on training sets are not an option, we argue that algorithms capable of such online analyses are superior regarding their applicability.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

We observed that evaluating log clustering approaches is far from trivial. In order to quantitatively determine the quality of the generated clusters, anomalies or patterns, ground truth consisting of labeled log data or at least expected signatures are required. Moreover, since log data collected from specific sources often exhibits peculiarities, proper evaluation should always be carried out on multiple datasets. However, generating labeled data usually requires time-consuming manual work. Thus, open-source labeled log data would be highly beneficial for objective comparisons and would allow researchers to benchmark approaches in a thorough manner. We were pleased to see that He et al. [HZZL17,HZH+17] not only provide the code of their algorithms, but also reimplement other approaches and further collect log datasets including labels [HZHL20] that specify which log lines belong to the same clusters, i.e., originate from the same log events. This enables reproducibility and proper comparisons among different approaches. We hope to see more researchers contributing or making their data and code accessible to the public.

Finally, we also point out that only few authors injected actual attacks in their datasets, but rather targeted failures and errors. While these could of course be artifacts of attacks, we suggest to use real attack scenarios in future evaluations. Since anomaly detection is applied in intrusion detection solutions, use cases more closely related to cyber threats have the potential to expand the possible application areas.

5.4 Summary

Log clustering plays an important role in many application fields, including cyber security. Accordingly, a large number of approaches have been developed in the past. However, existing approaches are often designed towards particular objectives, make specific assumptions on the log data and exhibit characteristics that prevent or foster their application in certain domains. This makes it difficult to select one or multiple approaches for a use case at hand. In this chapter we therefore carried out a survey that groups clustering approaches according to attributes that support such decisions.

As part of the survey, several gaps of state-of-the-art approaches for log clustering were identified, including a need for parser generation methods that leverage tree structures as well as tokenization on arbitrary characters. As a consequence, we proposed and developed a concept for a parser generator that ingests complex and unstructured log data and produces parsers that are immediately applicable for specific IDSs [WLSK19]. The resulting parsers are leveraged in the following chapter to process log data for anomaly detection, in particular, to extract categorical values of log events for correlation analysis.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 6

Anomaly Detection

The previous chapter outlined a survey for log clustering techniques. The findings of this survey contributed to the development of an approach for automatic parser generation that is presented in [WLSK19]. This approach enables the automatic generation of parsers for the log data presented in Chap. 3 and Chap. 4. In this chapter, an anomaly detection method is introduced that leverages parsed log events from these datasets for evaluation and validation. Major parts of this chapter have been published in [LHW+21].

As stated in Sect. 2.2.2, categorical variables are common in system logs and complement the detection of anomalous events. In particular, variables such as user identifiers, IP addresses, service names, system operations, or program states, occur in regular patterns that are expected to persist over time as long as the system behavior remains steady. For example, services utilize specific subsets of all available system operations and execute them with particular relative frequencies. Unexpected deviations from such conditional occurrence distributions indicate a change of system behavior and should therefore be reported to the system operators as anomalies. Unfortunately, the selection of variables suitable for such a detection mechanism is non-trivial, because it usually relies on expert knowledge about the system at hand and is difficult to automatize.

We propose the Variable Correlation Detector (VCD) as a solution to aforementioned issues. The approach comprises of a sequence of selection constraints to reduce the search space and identify interesting correlations between categorical variables. In addition, the VCD reuses conditional distributions of value occurrences computed in the selection phase for the disclosure of deviations in a subsequent detection phase. Our approach has several advantages over state-of-the-art methods. First, it identifies interesting correlations independent from the total occurrences of the involved values, which is different to approaches based on frequent itemset mining [TH19]. This is especially important for the detection of stealthy attacks that only produce infrequent values. Second, our approach does not generate strict rules for value co-occurrences, but instead involves fuzzy rules that do not always have to be fulfilled by employing statistical tests on chunks of events.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Third, our approach is designed for online detection in streams of log data, which is essential for application in real-world scenarios. An implementation of our approach is available online as part of the log-based anomaly detection system AMiner [AMi]. We summarize our contributions as follows:

- An iterative method for selecting useful correlations of categorical variables,
- an online anomaly detection technique based on identified correlations, and
- an evaluation of our open-source implementation of the proposed concepts.

The remainder of this chapter is structured as follows. In Sect. 6.1, we outline the concept of the VCD. We then provide details of our proposed correlation selection constraints in Sect. 6.2. We present the evaluation of our algorithm in Sect. 6.3 and discuss the results in Sect. 6.4. Finally, Sect. 6.5 summarizes the chapter.

6.1 Concept

This section outlines the concept of the Variable Correlation Detector (VCD). First, we explain important aspects of correlations of variables. Then, we state definitions relevant for this chapter and outline the overall procedure of the VCD.

6.1.1 Correlations of Variables

Log data are chronological sequences of events. Most log datasets comprise of a certain number of different event types, where each type defines the syntax of the corresponding log lines (cf. Sect. 5.1.1). Accordingly, simple log data such as comma-separated-values only consist of a single event. In any way, each event type specifies a sequence of variables or features. For example, the syscall event in Audit logs consists of a sequence of key-value pairs, such as “syscall=2” that specifies the syscall type or “items=1” that specifies the number of associated path records.

Some variables are strongly correlated, meaning that the occurrence of a value in one variable indicates the occurrence of a specific value in another variable. Given a sufficiently large time frame, these conditional probabilities should be more or less constant on a system with stable behavior. Any changes to these occurrence patterns indicate potentially malicious activities, i.e., anomalies.

Table 6.1 shows the number of occurrences of syscall types and items extracted from 10000 Audit logs that are also used in the evaluation in Sect. 6.3.1. With 7195 total occurrences, the majority of these events involve “items=0” (sum of first row). However, it is visible that syscall type 2 (“open”) mostly occurs with “items=1” (2592 occurrences) and sometimes “items=2” (90 occurrences), but never with “items=0”. Since other value pairs exhibit similar dependencies, it is reasonable to monitor the conditional probability

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Table 6.1: Value co-occurrences of syscall types and items in Audit logs

	syscall type									
items	0	1	2	20	42	49	59	90	105	**Total**
0	6097	860	0	189	34	14	0	0	1	7195
1	0	0	2592	0	104	0	0	5	0	2701
2	0	0	90	0	0	0	14	0	0	104

distributions of the variable "items" with respect to "syscall" for improved detection over monitoring the occurrences of "items" alone. The same reasoning applies for the other direction, i.e., monitoring the occurrences of syscall types given the number of items.

Different to existing approaches, we do not only focus on the selection of variables that are suitable for such correlations, but monitor the co-occurrences of their values. Thereby, we are not solely interested in frequent values or value combinations, but instead calculate the conditional probability distributions of all values that are useful for anomaly detection. Consider syscall type 59 ("exec") as an example: Even though the value only occurs in 14 events, it always co-occurs with "items=2" and thus indicates a strong correlation. Due to the large number of possible combinations of variables and distinct values, a brute-force solution is computationally not feasible in practice, especially for high-volume log data with diverse values. This chapter therefore presents an iterative selection strategy for interesting correlations that is presented in the following sections.

6.1.2 Definitions

As mentioned in the previous section, most log files comprise of several events $\mathcal{E}$, each containing a unique set of variables. For simplicity, we only consider a single event $E \in \mathcal{E}$ in the following and assume that the procedure is applicable to all other events analogously. Moreover, we assume that event E involves n variables $V_1, .., V_n$, each comprising of an arbitrary number of values $v_{i,1}, ..., v_{i,m_i}$ from the unique value set $\mathcal{V}_i$. We compute the estimated value occurrence probability as $P(v_{i,j}) = |\{V_i = v_{i,j}\}| / N$ in a sample of size N and the conditional probabilities as $P(v_{i,j} \mid v_{k,l}) = |\{V_i = v_{i,j} \wedge V_k = v_{k,l}\}| / |\{V_k = v_{k,l}\}|$. Correlations are denoted using the $\rightsquigarrow$ operator. Table 6.2 summarizes all symbol definitions.

6.1.3 Procedure

Our approach selects variable fields of log events and performs statistical tests on value occurrences in these fields for the purpose of anomaly detection. To limit the search space, we propose several sequential analysis steps that act as filters for all possible variable and value combinations. Figure 6.1 shows these steps as a state chart, with an in-depth description of each step following in Sects. 6.2.2-6.2.6.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 6.2: Definitions of symbols used in this chapter

Symbol	Definition
E	Log event type from the set of all event types $\mathcal{E}$, i.e., $E \in \mathcal{E}$.
V_i	Variable of log event type E, with $V_1, ..., V_n \in E$.
$\mathcal{V}_i$	Set of distinct values attained by V_i.
$v_{i,j}$	Value j of variable V_i, i.e., $\mathcal{V}_i = \{v_{i,1}, ..., v_{i,m_i}\}$.
$P(v_{i,j})$	Probability that value j occurs in V_i.
$P(v_{i,j} \mid v_{k,l})$	Probability that value j occurs in V_i given that value l occurs in V_k.
$V_i \rightsquigarrow V_k$	Correlation between variables V_i and V_k.
$v_{i,j} \rightsquigarrow v_{k,l}$	Correlation between values of variables, i.e., occurrence of value j in V_i correlates with value l of V_k.
θ_i	Threshold parameter for correlation selection.
N	Size of the sample for computing correlations during initialization.
M	Size of the sample for updating and testing in online mode.

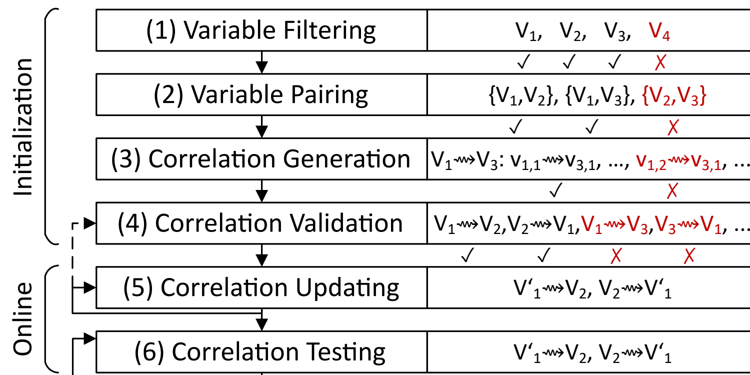


Figure 6.1: Procedure of the Variable Type Detector. Correlations between variables and values are filtered iteratively.

For the initialization phase in steps (1)-(4), the VCD first collects a sample of N log lines. We assume that all available variables of a log event are possible choices for correlations and that there is no manual pre-selection. Step (1) *Variable Filtering* sorts out variables that are unlikely to yield useful correlations, such as variables with many unique or static values. This step is exemplarily visualized in the figure by removing variable V_4 for subsequent analyses steps.

Step (2) *Variable Pairing* then generates pairs of the remaining variables V_1, V_2, V_3. This step removes pairs with dissimilar value probability distributions or disjoint value sets. In the figure, the variable pair $\{V_2, V_3\}$ is not considered for correlation. Remaining pairs are transformed to correlation hypotheses in step (3) *Correlation Generation*, where

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

conditional occurrence probabilities of all involved values are computed. Correlations between values denoted by $\rightsquigarrow$ that exhibit weak associations are omitted, e.g., values that occur in many combinations or have similar conditional probabilities to other correlated values. In the figure, value correlation $v_{1,2} \rightsquigarrow v_{3,1}$ of variable correlation $V_1 \rightsquigarrow V_3$ is removed. Note that correlations are directed, i.e., $V_1 \rightsquigarrow V_3$ is different from $V_3 \rightsquigarrow V_1$.

Step (4) *Correlation Validation* then evaluates whether all resulting value correlations indicate a sufficiently strong dependency between the correlated variables, in particular, whether the valid value correlations have independent probability distributions and involve sufficiently many occurring values. For example, assuming that several value correlations such as $v_{1,2} \rightsquigarrow v_{3,1}$ were removed in step (3), the variable correlation $V_1 \rightsquigarrow V_3$ is removed. This marks the end of the initialization phase, which is only executed once for every log event type.

For online anomaly detection, all correlation hypotheses that remain after step (4) are transformed into rules, which are repeatedly evaluated using samples of size M. For this, we perform statistical tests in step (5) *Correlation Updating* and go back to step (3) to re-initialize the correlation rules if value distributions change or new values appear, e.g., V_1 is replaced by V_1' in Fig. 6.1. Once correlation rules are stable for a sufficiently long time period and should not be updated anymore, they are tested in step (6) for the purpose of anomaly detection.

6.2 Approach

This section presents detailed explanations of all aforementioned steps of the VCD procedure. We also provide examples for the various selection criteria.

6.2.1 Sample data

We provide a small sample to make the equations in the following sections easier to understand and to obtain a rough estimate for reasonable choices for threshold parameters θ_i. The data shown in Table 6.3 comprises of one event with four variables, i.e., $E = \{V_1, V_2, V_3, V_4\}$, and a sample size of $N = 12$. We point out that this data is only for illustrative purposes and that the application of the VCD in practice requires sufficiently large sample sizes for appropriate probability estimation. Each variable involves three possible values, in particular, $\mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}_3 = \{1, 2, 3\}$. The occurrence probabilities of the values of V_1 are computed as $P(v_{1,1}) = \frac{7}{12}, P(v_{1,2}) = \frac{4}{12}, P(v_{1,3}) = \frac{1}{12}$. The data is set up so that V_1 and V_2 correlate, i.e., the occurrence of any value in V_1 usually co-occurs with the same value in V_2. This is also reflected in the conditional probabilities, e.g., $P(v_{2,1} \mid v_{1,1}) = \frac{5}{7}, P(v_{2,2} \mid v_{1,2}) = \frac{3}{4}$. On the other hand, V_3 and V_4 do not show a strong correlation with any other variable. Accordingly, the following examples usually set the thresholds θ_i so that correlations involving V_3 and V_4 are removed, but $V_1 \rightsquigarrow V_2$ and $V_2 \rightsquigarrow V_1$ are selected as relevant for detection.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Table 6.3: Sample data

ID	V_1	V_2	V_3	V_4
1	1	1	1	1
2	1	1	2	1
3	1	1	3	3
4	1	2	1	1
5	1	1	3	1
6	2	2	2	2
7	1	2	1	1
8	2	3	2	1
9	3	3	3	1
10	2	2	1	1
11	2	2	1	3
12	1	1	2	1

6.2.2 Variable Filtering

This section covers heuristics for variables. The first criterion targets variables with many unique values and the second criterion addresses dominating values.

Diversity of Values

Correlation analysis as it is done by the VCD requires categorical variables to reasonably calculate occurrence probabilities from the sample. Accordingly, we assume that there is a finite number of different values attained by each variable and that the sample size is large enough to obtain an estimate on their occurrence probabilities, i.e., $|\mathcal{V}_i| \ll N$. Variables with a large number of unique values are likely discrete rather than categorical, e.g., event IDs or timestamps, and do not yield stable correlations as described in Sect. 6.1.1. The reason for this is that they result in a high number of infrequent value co-occurrences that do not represent any actual correlation between the variables, e.g., an event ID is usually a random value. Equation 6.1 thus defines an upper limit for the number of unique values in V_i, where $\theta_1 \in [0, 1]$. From all available variables, we select all V_i that fulfill Eq. 6.1, and omit all others.

$$|\mathcal{V}_i| \leq \theta_1 \cdot N \tag{6.1}$$

The small sample size of the data in Table 6.3 requires $\theta_1 \geq 0.25$ to retain the variables, e.g., $\theta_1 = 0.25$ yields a critical value of 3 and $|\mathcal{V}_1| \leq 3$ is fulfilled.

Distribution Probabilities

In some variables, one or few values are occurring more often than others and are thus dominating the value probability distribution. These variables usually have weaker correlation with other variables, since most correlated values co-occur with the same dominating value. An extreme case of this situation are static variables, where the same value occurs in every log line and is thus trivially useless for correlation. We therefore use Eq. 6.2 to select only variables V_i where no occurrence probability of $v_{i,j}$ exceeds a certain limit. To allow more unique values $\theta_2 \in [0, 1]$ should be selected closer to 1.

TU Bibliothek — Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar. The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek. WIEN Your knowledge hub

$$P(v_{i,j}) < \theta_2 + \frac{1-\theta_2}{|\mathcal{V}_i|} \tag{6.2}$$

We point out that this heuristic causes that variables with similarly dominated value probability distributions that may have a strong association between the values are omitted. Since this heuristic is mainly used to efficiently limit the search space, it is possible to set θ_2 to a sufficiently large value to include these variables and use subsequent analysis steps to omit incorrect variable pairings.

The data from Table 6.3 involves value $v_{4,1}$ which dominates V_4. Setting $\theta_2 = 0.6$ excludes only this variable, since $P(v_{4,1}) = 0.75$ exceeds $0.6 + \frac{1-0.6}{3} = 0.73$.

6.2.3 Variable Pairing

This section describes criteria for selecting pairs of variables suitable for correlation. The first criterion matches variables with similar probability distributions and the second criterion addresses common value spaces.

Similarity of Distributions

As pointed out in the previous section, variables with similar value probability distributions are more likely to exhibit associations between their values than other variable pairs. The reason for this is that similar distributions imply that for each value in V_i there exists another value in V_k that occurs roughly the same amount of times and may thus have a direct relationship with the former value. On the other hand, comparing the value occurrences of one dominated distribution and another evenly distributed distribution, there is necessarily at least one value in one variable that co-occurs with more than one value in another variable, which indicates a weaker association.

We therefore generate variable pair $\{V_i, V_k\}$ if the occurrence probabilities $P(v_{i,j})$ of all values in V_i do not differ from $P(v_{k,l})$ in V_k, where each value is only used once. Equation 6.3 describes this rule formally, where $\theta_3 \in [0, \infty)$ and $p = 1, ..., min(|\mathcal{V}_i|, |\mathcal{V}_k|)$ is the index of the order statistic so that $v_{i,(1)}$ is the most occurring value of V_i, $v_{i,(2)}$ is the second most occurring value of V_i, etc.

$$\left|P\left(v_{i,(p)}\right) - P\left(v_{k,(p)}\right)\right| \leq \frac{\theta_3}{max(|\mathcal{V}_i|, |\mathcal{V}_k|)} \tag{6.3}$$

Setting $\theta_3 = 0.6$ yields a critical value of $\frac{0.6}{3} = 0.2$. In this case, variables V_1 and V_2 from Table 6.3 are correctly paired, since all probability differences $|P(v_{1,1}) - P(v_{2,1})| = 0.16$, $|P(v_{1,2}) - P(v_{2,2})| = 0.08$, and $|P(v_{1,3}) - P(v_{2,3})| = 0.08$ are lower than 0.2, where values are compared in decreasing order of their occurrences. Assuming that V_4 is not removed in the variable filtering phase, the pair $\{V_2, V_4\}$ is omitted since $|P(v_{2,1}) - P(v_{4,1})| = 0.33$ which exceeds 0.2.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

Common Values

Another heuristic is that variables sharing common values are likely related in some way. For example, log lines that involve separate variables for source and destination IP addresses often have the same value space, since data is sent and received from the same IP addresses. This also applies to state transitions in logs, such as network logs that contain messages like "inactive -> scanning", "scanning -> authenticating", etc. As an alternative in case that Eq. 6.3 is not fulfilled, we select pairs $\{V_i, V_k\}$ where both variables share a certain fraction of common values. This corresponds to selecting variable pairs that fulfill Eq. 6.4, where $\theta_4 \in [0, 1]$. For the sample data displayed in Table 6.3, this constraint is trivially fulfilled since all variables have the same value space.

$$|\mathcal{V}_i \cap \mathcal{V}_k| \geq \theta_4 \cdot min(|\mathcal{V}_i|, |\mathcal{V}_k|) \tag{6.4}$$

6.2.4 Correlation Generation

This section outlines the generation of correlation hypotheses for values of variable pairs. Note that each pair $\{V_i, V_k\}$ is considered as the two hypotheses $V_i \rightsquigarrow V_k$ and $V_k \rightsquigarrow V_i$ that are analyzed separately.

Diversity of Correlations

For optimal variable correlation, each value of one variable only occurs with a particular value of another variable and vice versa. Conversely, values that co-occur with many different values from the correlated variable indicate weak or random associations as pointed out in Sect. 6.2.2 and should not be considered for correlation hypotheses. We therefore select only value correlations $v_{i,j} \rightsquigarrow v_{k,l}$ for hypothesis $V_i \rightsquigarrow V_k$ if the relative amount of co-occurring values of $v_{i,j}$ does not exceed $\theta_5 \in [0, 1]$, i.e., if Eq. 6.5 is fulfilled.

$$\frac{|\{v_{k,l} : P(v_{k,l} \mid v_{i,j}) > 0\}|}{|\mathcal{V}_k|} \leq \theta_5 \tag{6.5}$$

Selecting $\theta_5 = 0.7$ for the data from Table 6.3 yields that $v_{1,1} \rightsquigarrow v_{2,l}$ of $V_1 \rightsquigarrow V_2$ are fulfilled for all l, since $v_{1,1}$ only occurs with $v_{2,1}, v_{2,2}$ and $\frac{2}{3} \leq \theta_5$. Similarly, $v_{1,2} \rightsquigarrow v_{2,l}$ yield $\frac{2}{3}$ and $v_{1,3} \rightsquigarrow v_{2,l}$ yield $\frac{1}{3}$, thus all possible value correlations from $V_1 \rightsquigarrow V_2$ are selected. On the other hand, all $v_{1,1} \rightsquigarrow v_{3,l}$ of $V_1 \rightsquigarrow V_3$ are omitted since $v_{1,1}$ co-occurs with three values in V_3 and $\frac{3}{3}$ exceeds θ_5.

Skewness of Distributions

If Eq. 6.5 from the previous section is not fulfilled, we use an alternative selection constraint to avoid that useful correlations are omitted too easily. In particular, we check the shape of the conditional distributions to identify dependencies between values, i.e., if one of the values in V_k occurs with relatively high frequency given that $v_{i,j}$ occurs, we

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

add $v_{i,j} \rightsquigarrow v_{k,l}, \forall l$ to the hypothesis $V_i \rightsquigarrow V_k$. Equation 6.7 shows that this constraint is realized by subtracting the highest from the lowest of all conditional probabilities given $v_{i,j}$ (cf. Eq. 6.6), where $\theta_6 \in [0, \infty)$. The idea behind this is that the difference is large for skewed distributions where some values co-occur frequently and others only rarely, and small for evenly distributed values. Note that this does not take into consideration that dominating values in V_k could incorrectly cause that the constraint is fulfilled, which is addressed in the following section.

$$\mathcal{P}_{i,j,k} = \{P(v_{k,l} \mid v_{i,j}) : P(v_{k,l} \mid v_{i,j}) > 0, \forall l\} \tag{6.6}$$

$$\max(\mathcal{P}_{i,j,k}) - \min(\mathcal{P}_{i,j,k}) > \frac{\theta_6}{|\{v_{k,l} : P(v_{k,l} \mid v_{i,j}) > 0\}|} \tag{6.7}$$

We use $\theta_6 = 0.8$ as a sample for the data in Table 6.3 and assume that $v_{1,2} \rightsquigarrow v_{2,l}, \forall l$ was omitted by the constraint from Eq. 6.5. Then $P(v_{2,1} \mid v_{1,1}) - P(v_{2,2} \mid v_{1,1}) = 0.42$ and $P(v_{2,2} \mid v_{1,2}) - P(v_{2,3} \mid v_{1,2}) = 0.5$ both exceed the critical value of $\frac{0.8}{2} = 0.4$. However, $v_{1,1} \rightsquigarrow v_{3,l}, \forall l$ is not fulfilled, because $P(v_{3,1} \mid v_{1,1}) - P(v_{3,3} \mid v_{1,1}) = 0.14$ does not exceed the critical value of $\frac{0.8}{3} = 0.27$ and is therefore correctly omitted from hypothesis $V_1 \rightsquigarrow V_3$.

6.2.5 Validation of Correlations

This section presents hypothesis validation constraints that omit correlations without sufficiently strong dependencies between values or few correlating values.

Dependencies of Distributions

As pointed out in Sect. 6.2.4, a valid correlation $V_i \rightsquigarrow V_k$ should imply that the conditional value probabilities $P(v_{k,l} \mid v_{i,j})$ differ from each other depending on the value $v_{i,j}$ attained by V_i. Otherwise, the values in V_k are independent from the attained values in V_i, which means that the correlation hypothesis should be discarded. We address this by measuring the variances of all conditional distributions in V_k with respect to the overall distribution of V_k. Equation 6.8 shows that the variances are added for all value correlations selected by one of the constraints from Sect. 6.2.4. Since variances of more frequently occurring value correlations are more representative for the variable and should therefore have a higher influence on the result, Eq. 6.9 with $\theta_7 \in [0, \infty)$ weights all variances by the occurrence probabilities of $v_{i,j}$ and checks whether their sum exceeds a threshold. In this case, the conditional distributions involved in the correlation hypothesis are sufficiently dependent and thus selected, otherwise the correlation is omitted from further analysis.

$$\mathbb{V}_k(v_{i,j}) = \sum_l \left\{ (P(v_{k,l} \mid v_{i,j}) - P(v_{k,l}))^2 : v_{i,j} \rightsquigarrow v_{k,l} \right\} \tag{6.8}$$

$$\sum_j \{\mathbb{V}_k(v_{i,j}) \cdot P(v_{i,j}) : v_{i,j} \rightsquigarrow v_{k,l}\} \geq \theta_7 \tag{6.9}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

We first consider correlation $V_1 \rightsquigarrow V_2$ from Table 6.3 as an example and use $\theta_7 = 0.2$ as a threshold. The variances $\mathbb{V}_2(v_{1,1}) = 0.13$, $\mathbb{V}_2(v_{1,2}) = 0.29$, and $\mathbb{V}_2(v_{1,3}) = 1.04$ are weighted by probabilities $P(v_{1,1}) = 0.58$, $P(v_{1,2}) = 0.33$, and $P(v_{1,3}) = 0.08$ respectively to yield a total of 0.26 that exceeds $\theta_7 = 0.2$. Accordingly, the conditional value distributions in V_2 sufficiently depend on the attained values in V_1, thus $V_1 \rightsquigarrow V_2$ is selected as a valid correlation. On the other hand, the weighted sum of variances for $V_3 \rightsquigarrow V_1$ yields 0.06, which does not exceed the threshold and thus indicates that the correlation should be omitted.

Value Coverage

The second selection criterion for value correlations from one of the constraints from Sect. 6.2.4 ensures that only variable correlations supported by sufficiently many correlating values are selected. In other words, a correlation $V_i \rightsquigarrow V_k$ is omitted if only a small fraction of the values in V_i have corresponding correlations. Thereby, we use the occurrence probabilities of $v_{i,j}$ to weight frequent values higher. According to Eq. 6.10, we only select $V_i \rightsquigarrow V_k$ if the relative amount of correlating values exceeds a threshold $\theta_8 \in [0, 1]$.

$$\sum_j \{P(v_{i,j}) : v_{i,j} \rightsquigarrow v_{k,l}\} \geq \theta_8 \tag{6.10}$$

We use data from Table 6.3 and consider the variable correlation $V_1 \rightsquigarrow V_3$ with $\theta_8 = 0.5$. We assume that all correlations from $v_{1,1}$ to values from V_3 were removed as outlined in the example in Sect. 6.2.4, but correlations from $v_{1,2}$ and $v_{1,3}$ to V_3 exist. The sum of probabilities for these values is then $P(v_{1,2}) + P(v_{1,3}) = 0.416$. Since this sum does not exceed the threshold of 0.5, correlation $V_1 \rightsquigarrow V_3$ is omitted from further analysis. Assuming that all value correlations were selected for $V_1 \rightsquigarrow V_2$ the constraint is trivially fulfilled since the sum of all occurrence probabilities always equals 1 and thus exceeds the threshold.

6.2.6 Correlation Updating and Testing

The previous sections outlined the initialization phase of the VCD, where correlations are selected by a sample of N log lines. Afterwards, the VCD switches to online mode, where samples of M log lines are repeatedly collected and tested with respect to the previously generated correlation rules. In the following, we use $\widetilde{P}$ to denote occurrence probabilities of values from these test samples. We use a two-sample Chi-squared test for homogeneity [BJS+11] to determine whether a test sample corresponds to the rules. For this, we first compute a test statistic t by comparing the conditional probabilities of the training and test samples with the expected probability P_e based on the mean as shown in Eq. 6.11 and Eq. 6.12.

$$P_e = \frac{N \cdot P(v_{k,l} \mid v_{i,j}) + M \cdot \widetilde{P}(v_{k,l} \mid v_{i,j})}{N + M} \tag{6.11}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$$t = \sum_{l} \left(N \cdot \frac{\left(P\left(v_{k,l} \mid v_{i,j}\right) - P_e\right)^2}{P_e} + M \cdot \frac{\left(\widetilde{P}\left(v_{k,l} \mid v_{i,j}\right) - P_e\right)^2}{P_e} \right) \tag{6.12}$$

For a given $v_{i,j}$, we then define an indicator function $I_k(v_{i,j})$ in Eq. 6.13 that is 1 if the test statistic does not exceed a critical value given by the Chi-squared distribution with confidence $\alpha_1 \in [0, 1]$, i.e., there is no significant difference between the conditional distributions of the training and test samples, and is 0 otherwise. We then store these indicators for all $v_{i,j} \in \mathcal{V}_i$ in a list $r_{i,j}$, so that $r_{i,j}^{(t-d)}, ..., r_{i,j}^{(t)}$ are the d most recent indicators after t tests of $v_{i,j}$, and compute another test statistic $s_{i,j}^t = \sum_{x=t-d}^{t} r_{i,j}^{(x)}$ on these values. The purpose of this is to reduce the number of false positives, i.e., anomalies are only reported when a certain number of Chi-squared tests fail. Since r is a binomial process, we use Eq. 6.14 to compute a critical value λ, where $\alpha_2 \in [0, 1]$ is the confidence of the binomial test and α_1 is reused as the success probability of the Chi-squared test. If $s_{i,j}^t \geq \lambda$ holds, there is no significant change of the conditional probabilities of $v_{i,j} \rightsquigarrow v_{k,l}, \forall l$ at test t, and vice versa. Note that the runtime can be reduced by computing λ a single time in advance when d, α_1, and α_2 remain constant.

$$I_k(v_{i,j}) = \begin{cases} 1 & \text{if} \quad t < \chi^2_{\alpha_1, |\mathcal{V}_k|-1} \\ 0 & \text{otherwise} \end{cases} \tag{6.13}$$

$$\lambda = min \left\{ k_{max} : \sum_{k=0}^{k_{max}} \frac{d!}{k! \cdot (d-k)!} \cdot \alpha_1^k \left(1 - \alpha_1\right)^{d-k} > 1 - \alpha_2 \right\} \tag{6.14}$$

The aforementioned computations are carried out for updating as well as testing correlations. The main difference between both phases is that step (6) *Correlation Testing* only reports anomalies when tests fail, i.e., $s < \lambda$, meaning that all changes of correlations are reported every time after processing the test samples as long as they persist. On the other hand, step (5) *Correlation Updating* adjusts the base line for comparison by updating distributions with newly observed values, removing correlations if the binomial test fails, and periodically repeating steps (3)-(4) to identify new correlations. Accordingly, this phase is seen as an extended training phase that is essential for online learning.

6.3 Evaluation

This section outlines the evaluation of our approach. We first compare variable correlations selected from a real dataset with two well-known correlation metrics. We then showcase the detection capabilities of the VCD. Finally, we investigate the influences of thresholds on the selected variables based on properties of the input data.

6.3.1 Comparison with Association Metrics

This section compares the selected correlations of the VCD with well-known association metrics. We first describe the data and then show the results.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Data

We use 10000 Audit logs of type syscall from the publicly available AIT-LDSv1.1 [LSW+20a] that was presented in Chap. 3 for this evaluation. We select Audit logs, because they are a common source for log analysis and are sufficiently complex to be representative for all kinds of log data. In addition, they contain many categorical variables that correlate with varying strength and have diverse value occurrence distributions. Out of all 27 variables, we remove the timestamp as well as 6 static variables that only attain one value, because it is not possible to generate useful correlations with them. The VCD always omits these variables due to Eq. 6.1 and Eq. 6.2.

We use the remaining 20 variables to generate all 380 possible variable pairs and measure their association strength. For this, we employ two association metrics for nominal values with arbitrary many categories, (i) the Uncertainty Coefficient U [PTVF07] based on conditional entropy, and (ii) the Unbiased Tschuprow's T [Ber13] based on the Pearson Chi-squared statistic. Both metrics are in the range $[0, 1]$, where 0 indicates no association between the variables, and 1 indicates the highest possible dependency. However, while T is symmetrical, U is non-symmetrical and measures how well the dependent variable is predictable by the given variable. For example, the data in Table 6.1 yields $T(\{\text{syscall}, \text{items}\}) = 0.53$ as well as $U(\text{syscall} \mid \text{items}) = 0.58$ and $U(\text{items} \mid \text{syscall}) = 0.93$.

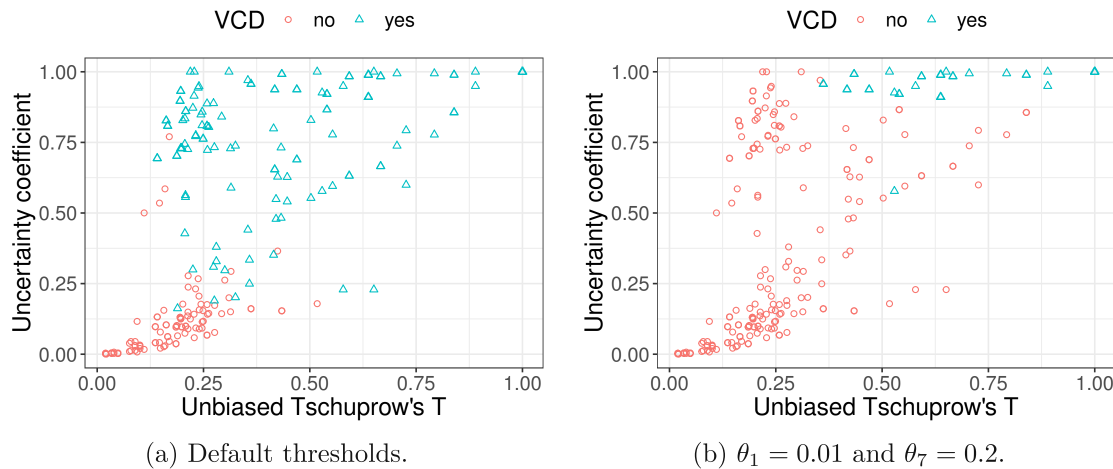
Results

We run the sequential selection steps of the VCD on the raw event logs and analyze the variable correlations that remain after the initialization phase. In Figs. 6.2a and 6.2b, these remaining correlations are marked "yes" (blue triangles), while all variable pairs that are omitted by one of the selection constraints are marked "no" (red circles). Each point in the scatter plots represents one of the 380 variable pairs displayed by their respective U and T, i.e., points closer to the top right corner of the plot indicate stronger association between the two involved variables, while points closer to the bottom left indicate weaker association.

The VCD was used with default settings (cf. Sect. 6.3.3) to classify the variable pairs in Fig. 6.2a. From all variable pairs, 222 were selected as interesting after the initialization phase and 158 were omitted. Since all of the omitted pairs received a relatively low association score by at least one of the metrics, we conclude that the VCD achieved to correctly omit irrelevant correlations. For example, among the omitted correlations is "syscall" $\rightsquigarrow$ "pid", which is reasonable as the process id "pid" is mostly random and independent from syscall types.

It is possible to further narrow down the set of tracked variable correlations by adjusting the thresholds. In particular, some of the variables involve large numbers of distinct values, which means that the number of monitored value correlations for pairs of these variables is immense. The default value $\theta_1 = 0.3$ allows 3.000 unique values in each variable, which is limited to 100 by setting $\theta_1 = 0.01$. This causes that the number of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



(a) Default thresholds.

(b) $\theta_1 = 0.01$ and $\theta_7 = 0.2$.

Figure 6.2: Comparison of VCD selection with association metrics.

remaining correlations drops from 222 to 126, where most of the rejected pairs are located close to the top left corner of the plot. Closer examination of these rejected pairs shows that they involve variables with many distinct values on the left side of the correlation and thus achieve a high U score, e.g., syscall arguments such as "a1" $\rightsquigarrow$ "items" with around 1000 unique "a1" values. Since their prediction strengths merely emerge from the large value space, adjusting θ_1 successfully omits these correlations.

In addition to adjusting θ_1, we increase θ_7 from 0.05 to 0.2 in Fig. 6.2b so that only variable pairs with strong dependency remain. This further reduces the amount of monitored correlations to 97 and omits correlations involving IDs such as "ppid" $\rightsquigarrow$ "exe", while more interesting correlations such as the sample correlation between "syscall" and "items" from Table 6.1 remains in both directions. We conclude that these experiments show the VCD is capable of selecting useful and strong correlations based on user-defined thresholds.

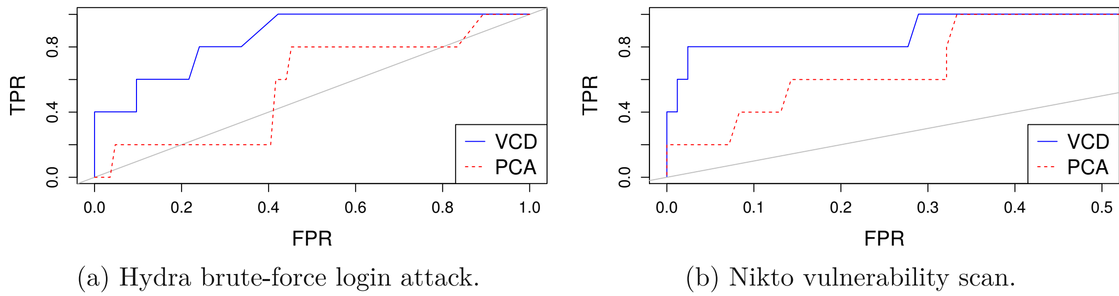
6.3.2 Anomaly Detection

This part of the evaluation validates the anomaly detection capabilities of the VCD. We first provide information on the log data and then present the results.

Data

We use Apache access logs from the AIT-LDSv1.1 [LSW$^+$20a] that was presented in Chap. 3 for this part of the evaluation. These logs are relevant, because they involve several categorical variables, including IP addresses, request methods (e.g., "GET", "POST"), resource names, status codes, etc. In addition, web-based attacks frequently manifest themselves as changes of multiple sequential values in these variables. In particular, we select (i) a brute-force login attack using Hydra [Hyd] that repeatedly requests the login web page with arbitrary user data, and (ii) a Nikto vulnerability scan [Nik] that requests

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) Hydra brute-force login attack.

(b) Nikto vulnerability scan.

Figure 6.3: Anomaly detection ROC plots for two attack scenarios.

non-available resources and thereby causes multiple redirects that correspond to status code 302. To evaluate detection accuracy with respect to different attack executions, we simulate varying intensities by injecting only a certain amount of events at particular times. Precisely, we inject batches of 5, 10, 20, 50, and 200 events for each attack in intervals of 10000 lines (around 12 hours). We label log line samples containing these batches as anomalous to measure the detection accuracy of the VCD in the following.

Results

For both attack cases, we configure the VCD to use the first $N = 10000$ lines of the Apache Access log files for initialization of the correlations. Thereby, we set $\theta_3 = 0.7$ and $\theta_7 = 0.005$ since the involved variables usually have different distributions and are relatively independent. All other parameters are used with default values (cf. Sect. 6.3.3). After initialization, we use a test sample size of $M = 1000$ to update the correlations on the remaining lines of the first day (20000 lines) using empirically determined confidences $\alpha_1 = 0.001$ and $\alpha_2 = 0.05$, and an indicator list size $d = 30$. This phase omits correlations that appear interesting during initialization, but are too unstable for anomaly detection. With the beginning of the second day, we switch the VCD from updating to testing mode, i.e., correlations that fail tests are no longer changed or omitted. We experiment with different values for α_1 in the test phase and count true positives (TP) as detected samples containing injected lines, false positives (FP) as detected normal samples, false negatives (FN) as undetected samples containing injected lines, and true negatives (TN) as undetected normal samples.

For comparison, we select Principal Component Analysis (PCA) as a baseline, because it allows to handle categorical data through one-hot encoding of values. Similar to the VCD, we use samples of 1000 lines to generate value count vectors and use the first 30000 lines for model building. In the subsequent detection phase, we measure the squared prediction error of test samples and mark them as anomalies if the error exceeds threshold Q_α, where confidence α is varied [HZHL16].

Figure 6.3a shows the trade-off between true positive rate ($TPR = \frac{TP}{TP+FN}$) and false positive rate ($FPR = \frac{FP}{FP+TN}$) of VCD and PCA in the first attack scenario. The results indicate that the VCD successfully detects the attack and yields $TPR = 60\%$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

(corresponding to the detection of the samples containing 20, 50, and 200 injected lines) at only $FPR = 10\%$. Closer inspection of the anomalies shows that involved variables are mainly "request" and "referer". In the training phase, the request to the login page "/login.php" occurs in 1.2% of all lines, half of these times with referer "http://mail.insect.com/login.php" and with "-" otherwise. However, requests to the login page made by the Hydra attack always have referer "-" and thus distort this distribution within the test sample, which is detected by the VCD. On the other hand, the PCA ROC curve indicates that it is only slightly better than random guessing. The reason for this is that the one-hot encoded data becomes very high-dimensional and PCA is thus unable to detect slight changes of single values in such complex models.

For the second attack scenario, relevant variables include the request method, where values "GET", "POST", and "OPTIONS" occur with 74%, 21%, and 5% in the training data respectively, as well as the status code, where 200 occurs in 96% and 302 in 4% of these lines. The Nikto scan generates lines with request method "GET" and status code "302", a combination that only occurs in 0.5% of all lines. Since the VCD is better suited to detect changes of occurrences conditioned by infrequent values such as "302"⇝"GET" of correlation "status"⇝"method", it performs better than PCA as visible in Fig. 6.3b.

6.3.3 Threshold Parameter Selection

The filtering steps for correlations between variables and values presented in Sect. 6.2 make use of threshold parameters θ_1-θ_8 to narrow down the search space and select only those correlations that are likely to positively contribute to the detection of anomalies. This section investigates the influence of these threshold parameters on the resulting correlations and thereby supports the manual parameter selection process, in particular, by relating each parameter to specific properties of the data at hand. In the following, we first explain the generation of synthetic data for this evaluation and then describe our experiments.

Data

To measure the influence of thresholds on the correlation selection, it is necessary to control properties of the input data. Therefore, we generate synthetic data for our experiments. We use three variables V_1, V_2, and V_3, of which only V_1 and V_2 correlate with varying strength, and monitor the correlations found by the VCD for different threshold settings. We use values $\mathcal{V}_i = \{0, 1, ..., x\}, x \in \mathbb{N}$ for each variable and compute their occurrence probabilities as normalized geometric series. Equation 6.15 shows how the probabilities for values in V_1 and V_3 are computed, where $p_i = 1$ means that all values are equally likely to occur, and lower values mean that one or more values are dominating the probability distribution. Equation 6.16 shows how the conditional probabilities of values in V_2 given values from V_1 are computed. Thereby, ρ specifies the correlation strength, i.e., larger values for ρ indicate that the same values co-occur more frequently with each other, and ζ is a damping factor that reduces the correlation strength for larger $v_{i,j}$, i.e., higher values for ζ cause more co-occurrences between different values.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

V_2 \ V_1	0	1	2	3	4	5	6	7	8	9
9	0	0	0	1	0	0	2	3	16	82
8	0	0	0	0	0	3	7	28	138	9
7	0	0	0	0	3	8	31	190	14	6
6	0	0	0	0	2	38	274	27	4	4
5	0	0	3	7	53	406	22	2	3	6
4	0	5	7	91	582	45	3	5	1	3
3	5	24	115	831	57	5	0	1	3	2
2	23	167	1260	81	4	1	1	4	1	6
1	292	1766	114	15	1	0	1	1	4	2
0	2852	193	17	1	0	2	0	3	2	9

Figure 6.4: Value co-occurrences of damped correlation.

$$P\left(v_{i,j}\right) = \frac{p_i^j}{\sum_{j'=0}^{|\mathcal{V}_i|} p_i^{j'}} \tag{6.15}$$

$$P\left(v_{k,l} \mid v_{i,j}\right) = \frac{(1-\rho)^{|j-l|} + \zeta^{||\mathcal{V}_i|-j|}}{\sum_{l'=0}^{|\mathcal{V}_k|} (1-\rho)^{|j-l'|} + \zeta^{||\mathcal{V}_i|-j|}} \tag{6.16}$$

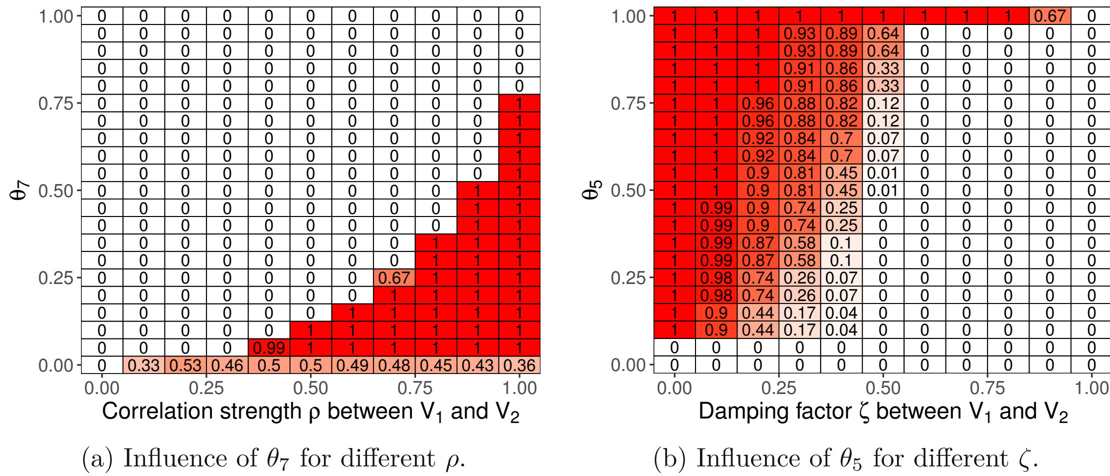
Figure 6.4 shows the co-occurrences of values from V_1 and V_2 for a sample configuration of $x = 9$, $p_1 = 0.7$, $\rho = 0.9$, and $\zeta = 0.4$. Due to the relatively strong correlation factor, most values in V_1 occur with the same value of V_2. The figure also shows that higher values of V_1 co-occur with more values of V_2 due to the damping factor, e.g., while $v_{1,1}$ only occurs with four different values of V_2, $v_{1,9}$ occurs with each value of V_2 at least once.

To evaluate the accuracy of the correlation selection procedure, we generate a ground truth of expected value correlations that contains all $v_{1,j} \rightsquigarrow v_{2,l}$ and $v_{2,l} \rightsquigarrow v_{1,j}$ that occur at least once in the data. We count correlations selected by the VCD and present in the ground truth as true positives (TP), correlations not present in the ground truth as false positives (FP), correlations missed by the VCD as false negatives (FN), and all other correlations as true negatives (TN). We use the F-score $F_1 = TP/\left(TP + 0.5 \cdot (FP + FN)\right)$ to measure the accuracy in the next section.

Results

We first experiment with θ_7, which is essential for selecting correlations that represent actual dependencies between the values and do not spuriously emerge from skewed value probability distributions. To analyze the relationship between θ_7 and the correlation strength, we increase θ_7 in steps of 0.05 and ρ in steps of 0.1 in the range $[0, 1]$ while leaving $p_1 = 0.7, p_3 = 0.7, \zeta = 0.4$ constant, generate 10 data samples with 10000 events respectively as outlined in the previous section, and then compute the average F-score of these simulation runs. The results visualized in Fig. 6.5a show that weaker correlation strengths require θ_7 to be sufficiently low to select all correct correlations and achieve the

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Influence of θ_7 for different ρ.

(b) Influence of θ_5 for different ζ.

Figure 6.5: Influence of thresholds on accuracy of correlation selection.

highest possible F-score of 1. However, setting θ_7 to 0 causes a decrease of the F-score independent of the correlation strength. The reason for this is that correlations involving V_3 are not checked for dependency and are thus incorrectly selected, which increases the number of FP. We therefore conclude that θ_7 should be set to a low, but non-zero value, e.g., 0.05. Note that the selection of θ_7 is not affected by ζ, since additional value co-occurrences only have little influence on the sum of variances as long as they are not dominating the distribution.

Threshold θ_5 on the other hand relies on the total number of co-occurrences for a given value and is thus influenced by ζ in addition to ρ. Figure 6.5b shows the F-score for various combinations of θ_5 and ζ, while $\rho = 1$ is fixed. As expected, increasing values for ζ yield lower F-scores for a given θ_5, because the number of distinct co-occurring values for any given value increases quickly (cf. Fig. 6.4). Accordingly, it is necessary to set $\theta_5 \geq 1$ for $\zeta > 0.5$ to select any correlations. For $\zeta \leq 0.5$, θ_5 effectively steers the allowed number of distinct co-occurrences, e.g., for $\theta_5 = 0.5$ at most 5 co-occurring values are allowed since $|\mathcal{V}_i| = 10, \forall i$.

We argue that the influence of other thresholds is trivial and therefore omit the plots for brevity. Table 6.4 shows a summary of all thresholds and the data properties with the highest influence on their selection. Note that θ_8 is most influenced by θ_5 and θ_6 rather than a property of the input data, because these thresholds regulate the generation of value correlations that affect the selection criterion involving θ_8. The table also provides default values that we identified as useful during our experiments and are used in the evaluations in Sect. 6.3.

These results indicate that the large number of parameters does not impede practical application of the VCD, since the thresholds are mostly independent from each other and allow to configure the correlation selection constraints specifically to counteract otherwise problematic properties of the data. For example, a high number of correlations

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Table 6.4: Dependencies and default values of thresholds

Thresh.	Infl. by	Default	Thresh.	Infl. by	Default
θ_1	$\lvert\mathcal{V}\rvert, N$	0.3	θ_5	ρ, ζ	0.5
θ_2	p	0.4	θ_6	ρ, ζ	1
θ_3	p, ρ	0.5	θ_7	ρ	0.05
θ_4	$\mathcal{V}$	0	θ_8	θ_5, θ_6	0.7

involving many distinct values (i.e., $|\mathcal{V}|$ is large) or weakly correlated variables (i.e., ρ is low) should be addressed by adjusting θ_1 and θ_7 accordingly to reduce the total number of correlations that are considered for anomaly detection as shown in Sect. 6.3.1.

6.4 Discussion

The evaluation in the previous section ascertains that the VCD selects appropriate variables for correlation analysis and detects anomalies by monitoring co-occurrences of correlated values over time. Thereby, the VCD makes use of a sequence of filtering steps that are separately configured by thresholds. We recognize that such a large number of parameters usually complicates practical application [TH19], however, we argue that this is not the case for the VCD since the thresholds are set relatively independent and specific to certain properties of the data (cf. Sect. 6.3.3). In addition, it is possible to omit single selection steps and iteratively refine the limits of the search space as we show in Sect. 6.3.1.

Our approach focuses on the correlation between pairs of variables rather than correlations where more than two variables are involved, e.g., $V_1 \rightsquigarrow \{V_2, V_3\}$ or $\{V_1, V_2\} \rightsquigarrow V_3$. However, we argue that this is trivial to achieve, since our selection criteria work analogously with combined occurrences of values. In fact, our implementation [AMi] supports correlation analysis of specific subsets of variables.

Finally, we suggest to develop selection strategies similar to the one presented in this chapter, but with a focus on mixes of categorical and continuous variables, i.e., categorical values indicate that values of another variable origin from a particular continuous distribution. For example, logged measurement data such as memory usage could follow a normal distribution with mean and variance depending on an active user. We leave this task for future research.

6.5 Summary

This chapter presents the Variable Correlation Detector (VCD), a novel approach for anomaly detection based on value co-occurrences in categorical variables of log events. The VCD comprises two modes. First, an initialization mode where variable and value

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

correlations are iteratively selected by multiple factors, such as skewness, similarity, and dependency of value occurrence probabilities as well as diversity and coverage of values. Second, an online learning and detection mode that continuously updates the identified correlations and reports anomalies based on deviations of the conditional occurrences. Other than state-of-the-art approaches, the VCD also analyzes infrequent values and recognizes system behavior changes that occur over long time spans.

As all anomaly-based detectors, application of the VCD in real-world scenarios is constrained by high amounts of false positives. To some degree, this situation may be related to inappropriate configurations and could thus be alleviated by modifying the thresholds accordingly. However, many anomalies may also be related to unusual but otherwise benign changes of the monitored system behavior, such as events caused by changes of the system landscape or erratic human behavior. In addition, both benign and malicious changes in behavior patterns are likely to raise multiple anomalies since correlations of categorical variables are often interrelated and thus collectively report anomalies. Whether they are true or false positives, analysts struggle to deal with such large numbers of anomalies. To alleviate this problem, the following chapter proposes an alert aggregation approach that groups anomalies and helps to identify false positives or specific attack cases.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

CHAPTER 7

CTI Extraction

The previous section outlined a technique for the detection of anomalous value correlations in log data. Anomalies reported by this detector provide some information on the expected and observed correlation behavior, but otherwise require human operators for interpretation and reasoning as the detector itself is unable to recognize specific attacks or assign corresponding labels. In addition, as all anomaly detection techniques, the presented approach may produce relatively high amounts of false positives. Alert aggregation pursues to alleviate these problems by grouping and correlating alerts into cyber threat intelligence (CTI), i.e., alert patterns that describe complex system behavior and enable automatic detection of similar attacks. This chapter therefore describes an alert aggregation technique that transforms anomalies generated by one or more detectors into higher-level CTI. Major parts of this chapter have been published in [LSWR22].

The situation where alerts occur in such large volumes that analysts are too overwhelmed to reliably assess and process them is commonly referred to as alert flooding [Jul03]. The number of produced alerts thereby usually depends on the deployed IDS as well as the type of attack. For example, attacks that result in many alerts include denial-of-service attacks that access machines with high intensity, brute-force attacks that repeatedly attempt to log into accounts with random passwords, and automatic scripts that search for vulnerabilities [HWF19]. These attacks produce high loads on the network and consequently cause the generation of many events in the monitored logs, of which a large part is reported by signature-based IDSs that search for patterns corresponding to such common attacks. On the other hand, anomaly-based IDSs such as the VCD presented in Chap. 6 that learn a baseline of normal system behavior and report alerts for statistical deviations are known to suffer from high false positive rates, i.e., they frequently report alerts during normal operation. Independent from their origin, alerts that occur in large frequencies are problematic, because they are difficult to categorize and may cause that analysts overlook other relevant alerts that occur less frequently [EO11, Jul03]. To

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

alleviate this issue, alerts should be filtered or aggregated before being presented to human analysts.

Alert aggregation techniques usually rely on automatic correlation or manual linking of alert attributes [NDP18]. However, organizations frequently deploy heterogeneous IDSs to enable broad and comprehensive protection against a wide variety of threats, causing that generated alerts have different formats and thus require normalization [SMFDV13]. Most commonly, attributes of alerts are thereby reduced to timestamps, source and destination IP addresses and ports, and IDS-specific classifications, which are considered the most relevant features of alerts [ASZ+16]. Unfortunately, alerts from host-based IDSs do not necessarily contain network information and alerts from anomaly-based IDSs do not involve alert types, which renders them unsuitable for existing aggregation techniques. In their survey, Navarro et al. [NDP18] therefore recommend to develop alert aggregation techniques that operate on general events rather than well-formatted alerts to avoid loss of context information. The authors also found that most existing approaches rely on predefined knowledge for linking alerts, which impedes detection of unknown attack scenarios. In addition, modern infrastructures consist of decentralized networks and container-based virtualization that prevent IP-based correlation [HWF19]. There is therefore a need for an automatic and domain-independent alert aggregation technique that operates on arbitrary formatted alerts and is capable of generating representative attack patterns independent from any pre-existing knowledge about attack scenarios.

IDSs generate streams of individual alerts. Aggregating these alerts means to group them so that all alerts in each group are related to the same root cause, i.e., a specific malicious action or attack. Unfortunately, finding such a mapping between alerts and attacks is difficult for a number of reasons. First, attack executions usually trigger the generation of multiple alerts [RSG10], because IDSs are set up to monitor various parts of a system and any malicious activity frequently affects multiple monitored services at the same time. This implies that it is necessary to map a set of alerts to a specific attack execution, not just a single alert instance. Second, it is possible that the same or similar alerts are generated as part of multiple different attacks, which implies that there is no unique mapping from alerts to attacks. This is caused by the fact that IDSs are usually configured for a very broad detection and do not only consist of precise rules that are specific to particular attacks. Third, repeated executions of the same attack do not necessarily manifest themselves in the same way, but rather involve different amounts of alerts and changes of their attributes. This effect is even more drastic when parameters of the attack are varied, their executions take place on different system environments, or alerts are obtained from differently configured IDSs. Fourth, randomly occurring false positives that make up a considerable part of all alerts [Jul03] as well as interleaving attacks complicate a correct separation of alerts that relate to the same root cause.

In addition, alert sequences should be aggregated to higher-level alert patterns to enable the classification of other alerts relating to the same root cause. In the following, we refer to these patterns as *meta-alerts*. The aforementioned problems are insufficiently solved by existing approaches, which usually rely on models built on pre-existing domain

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

knowledge, manually crafted links between alerts, and exploitation of well-structured alert formats.

This chapter thus presents a framework for automatic and domain-independent alert aggregation. The approach consists of an algorithm that groups alerts by their occurrence times, clusters these groups by similarity, and extracts commonalities to model meta-alerts. We summarize our contributions as follows:

- An approach for the incremental generation of meta-alerts from heterogeneous IDS alerts,
- similarity-metrics for semi-structured alerts and groups of such alerts,
- aggregation mechanisms for semi-structured alerts and groups of such alerts, and
- an evaluation of the proposed approach based on alerts from real-world systems.

The remainder of this chapter is structured as follows. Section 7.1 outlines important concepts of our approach, including alerts, alert groups, and meta-alerts. Section 7.2 describes the overall procedure of the framework. Section 7.3 explains the realization of the concepts with the aid of pseudo code. We present the evaluation of our approach in Sect. 7.4 and discuss the results. Finally, Sect. 7.5 summarizes the chapter.

7.1 Entities & Operations

This section presents relevant concepts of our alert aggregation approach. We first provide an overview of the entities and their relationships. We then discuss our notion of alerts, outline how alerts are clustered into groups, and introduce a meta-alert model based on aggregated alert groups.

7.1.1 Overview

Our approach transforms alerts generated by IDS into higher-level meta-alerts that represent specific attack patterns. Figure 7.1 shows an overview of the involved concepts. The top of the figure represents alerts occurring as sequences of events on two timelines, which represent different IDSs deployed in the same network infrastructure or even separate system environments. Another possibility is that events are retrieved from historic alert logs and used for forensic attack analysis.

Alert occurrences are marked with symbols and colors that represent their types. Thereby, two alerts could be of the same type if they share the same structure, were generated by the same rule in the IDS, or have coinciding classifications. We differentiate between square ($\square$), triangle ($\triangle, \triangledown, \triangleleft, \triangleright$), circle ($\circ$), and dash ($-$) symbols, which are marked blue, red, green, and yellow respectively. For the examples presented throughout this chapter, we consider alerts represented by one of $\{\triangle, \triangledown, \triangleleft, \triangleright\}$ as variations of the same alert type,

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

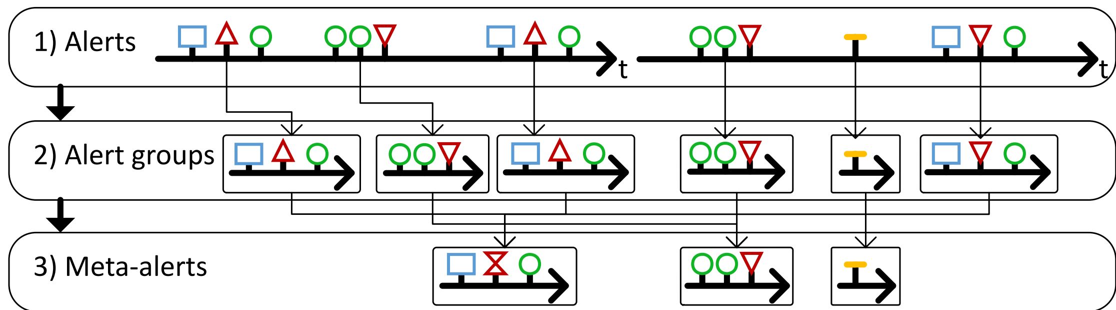


Figure 7.1: Overview of the relationships between concepts. Alerts (top) occurring on timelines (t) are grouped by temporal proximity (center) and then aggregated to meta-alerts by similarity (bottom).

i.e., these alerts have sufficiently many commonalities such as matching attributes and are thus similar to each other. In general, each alert represents a unique event that occurs only at one specific point in time. However, alerts of the same type, e.g., alerts that are generated by the same violation of a predefined rule or alerts reported by the same IDS, may occur multiple times. We mark these alerts accordingly with the same color.

As outlined in the beginning of this chapter, automatic mapping of alerts to higher-level meta-alerts is non-trivial. In the simple example shown in Fig. 7.1, it is easy to see that the alert sequence $(\square, \triangle, \circ)$ and the similar sequence $(\square, \triangledown, \circ)$ occur a total of three times, and that the pattern $(\circ, \circ, \triangledown)$ occurs two times over the two timelines. This is intuitively visible, because these alerts occur close together. Accordingly, it is reasonable to allocate alerts to groups that reflect this characteristic.

The center part of the figure shows groups of alerts based on their respective positions on the timelines. Note that grouping by alert type instead of temporal proximity would result in a loss of information, because alerts would be allocated to groups independent of their contexts, i.e., other alerts that are generated by the same root cause. For example, grouping all alerts of type $\circ$ would have neglected the fact that this type actually occurs in the patterns $(\square, \triangle, \circ)$ as well as $(\circ, \circ, \triangledown)$ and may thus not be a good indicator for a particular attack execution on its own.

Computing similarities between groups means measuring the differences of orders, frequencies, and attributes of their contained alerts. Alert groups that yield a high similarity are likely related to the same root cause and should thus be aggregated into a condensed form that reflects a typical instance of that group, i.e., a meta-alert. The bottom of the figure shows the generation of meta-alerts from similar groups. Thereby, orders, frequencies, and attributes of meta-alerts are created in a way to represent all allocated alert groups as accurate as possible. The figure shows that this is accomplished by merging the second alert in the patterns $(\square, \triangle, \circ)$ and $(\square, \triangledown, \circ)$ into alert ⧖, which combines attributes and values of $\triangle$ and $\triangledown$ so that both are adequately represented. In practice, this could mean that two different values of the same attribute in both alerts are combined into a set.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

The second meta-alert with alert sequence $(\circ, \circ, \triangledown)$ is formed from two identical groups and thus does not involve changes to merged alerts. If meta-alert generation was based on similarity of alerts rather than groups, all occurrences of similar alerts $\triangle$ and $\triangledown$ would be replaced with ⧖, thereby decreasing the specificity of the second meta-alert. This suggests that forming groups of logically related alerts is an essential step for meta-alert generation. Finally, the third meta-alert contains a single alert that only occurred once and is the only alert in its group. Since alerts form the basis of the presented approach, the following section will discuss their compositions in more detail.

7.1.2 Alerts

IDSs are designed to transmit as much useful information as possible to the person or system that receives, interprets, and acts upon the generated alerts. This includes data derived from the event that triggered the alert, e.g., IP addresses present in the monitored data, as well as information on the context of detection, e.g., detection rule identifiers. As outlined in Sect. 2.4, most existing approaches omit a lot of this information and only focus on specific predefined attributes. Our approach, however, utilizes all available data to generate meta-alerts without imposing any domain-specific restrictions.

To organize all data conveyed with each alert in an intuitive form, alerts are frequently represented as semi-structured objects, e.g., XML-formatted alerts as defined by the IDMEF [IDM] or JSON-formatted alerts generated by Wazuh [Waz] IDS. Even though such standards exist, different IDSs produce alerts with data fields specific to their detection techniques. For example, a signature-based detection approach usually provides information on the rule that triggered the alert, while anomaly-based IDSs only indicate suspicious event occurrences without any semantic interpretation of the observed activity. In addition, some IDSs do not provide all attributes required by standards such as IDMEF, e.g., host-based IDSs analyze system logs that do not necessarily contain network and IP information.

Figure 7.2 shows such an alert that was caused by a failed user login attempt generated by Wazuh. Note that it does not support IP-based correlation, since only "`srcip`" that points to localhost is available. The alert contains semi-structured elements, i.e., key-value pairs (e.g., "`timestamp`"), lists (e.g., "`groups`"), and nested objects (e.g., "`rule`"). In alignment with this observation, we model alerts as abstract objects with arbitrary numbers of attributes. Formally, given a set of alerts $\mathcal{A}$, an alert $a \in \mathcal{A}$ holds one or more attributes κ_a, where each attribute $a.k$ is defined as in Eq. 7.1.

$$a.k = v_1, v_2, ..., v_n \quad \forall k \in \kappa_a, n \in \mathbb{N} \tag{7.1}$$

Note that Eq. 7.1 also holds for nested attributes, i.e., $a.k.j, \forall j \in \kappa_{a.k}$, and that v_i is an arbitrary value, such as a number or character sequence. In the following we assume that the timestamp of the alert is stored in key $t \in \kappa_a, \forall a \in \mathcal{A}$, e.g., $a.t = 1$ for alert a that occurs at time step 1. These alert attributes are suitable to compare alerts and measure their similarities, e.g., alerts that share a high number of keys and additionally

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
{
        "timestamp": "2020-03-04T19:26:05.000000+0000",
        "rule": {
                "level": 5,
                "description": "PAM: User login failed.",
                "id": "5503",
                "firedtimes": 28,
                "groups": [
                        "pam",
                        "syslog",
                        "authentication_failed"
                ]
        },
        "full_log": "Mar  4 19:26:05 mail auth: pam_unix(dovecot:auth): authentication
            failure; logname= uid=0 euid=0 tty=dovecot ruser=daryl rhost=127.0.0.1
            user=daryl",
        "data": {
                "srcip": "127.0.0.1",
                "srcuser": "daryl",
                "dstuser": "daryl",
                "uid": "0",
                "euid": "0",
                "tty": "dovecot"
        },
        "location": "/var/log/forensic/auth.log"
}
```


Figure 7.2: Simplified sample alert documenting a failed user login.

have many coinciding values for each common key should yield a high similarity, because they are likely related to the same suspicious event. This also means that values such as IPs are not ignored, but matched by common keys like all other attributes. We define a function $alert_sim$ in Eq. 7.2 that computes the similarity of alerts $a, b \in \mathcal{A}$.

$$alert_sim : a, b \in \mathcal{A} \rightarrow [0, 1] \tag{7.2}$$

Thereby, the similarity between any non-empty alert and itself is 1 and the similarity to an empty object is 0. Furthermore, the function is symmetric, which is intuitively reasonable when comparing alerts on the same level of abstraction. On the other hand, the function implicitly computes how well one alert is represented by another more abstract alert as we will outline in Sect. 7.3.1. We summarize the properties of the function in Eq. 7.3-7.5.

$$alert_sim(a, a) = 1 \tag{7.3}$$

$$alert_sim(a, \emptyset) = 0, \quad a \neq \emptyset \tag{7.4}$$

$$alert_sim(a, b) = alert_sim(b, a) \tag{7.5}$$

As mentioned, we do not make any restrictions on the attributes of alerts and only consider the timestamp $a.t$ of alert a as mandatory, which is not a limitation since the time of detection is always known by the IDS or can be extracted from the monitored data. In the next section, this timestamp will be used to allocate alerts that occur in close temporal proximity to groups.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.1.3 Alert Groups

Alerts generated by an arbitrary number of deployed IDSs result in a sequence of heterogeneous events. Since attacks typically manifest themselves in multiple mutually dependent alerts rather than singular events, it is beneficial to find groups of alerts that were generated by the same root cause as shown in Sect. 7.1.1. In the following, we describe our strategies for formation and representation of alert groups that enable group similarity computation.

Formation

Depending on the type of IDS, alerts may already contain some kind of classification provided by their detection rules. For example, the message "`PAM: User login failed.`" contained in the alert shown in Fig. 7.2 could be used to classify and group every event caused by invalid logins. While existing approaches commonly perform clustering on such pre-classifications of IDSs, single alerts are usually not sufficient to differentiate between specific types of attacks or accurately filter out false positives. To alleviate this problem, we identify multiple alerts that are generated in close temporal proximity and whose combined occurrence is a better indicator for a specific attack execution. For example, a large number of alerts classified as failed user login attempts that occur in a short period of time and in combination with a suspicious user agent could be an indicator for a brute-force password guessing attack executed through a particular tool. Such a reasoning would not be possible when all alerts are analyzed individually, because single failed logins may be false positives and the specific user agent could also be part of other attack scenarios.

The problem of insufficient classification is even more drastic when alerts are received from anomaly-based IDS, because they mainly disclose unknown attacks. Accordingly, an approach that relies on clustering by alert classification attributes would require human analysts who interpret the root causes and assign a classifier to each alert. Temporal grouping on the other hand is always possible for sequentially incoming alerts and does not rely on the presence of alert attributes.

Our strategy for alert group formation is based on the interval times between alerts. In particular, two alerts $a, b \in \mathcal{A}$ that occur at times $a.t, b.t$ have an interval time $|a.t - b.t|$ and are allocated to the same group if $|a.t - b.t| \leq \delta$, where $\delta \in \mathbb{R}^+$. This is achieved through single-linkage clustering [ELLS11]. In particular, all alerts are initially contained in their own sets, i.e., $s_{\delta,i} = \{a_i\}, \forall a_i \in \mathcal{A}$. Then, clusters are iteratively formed by repeatedly merging the two sets with the shortest interval time $d = \min(|a_i.t - a_j.t|), \forall a_i \in s_{\delta,i}, \forall a_j \in s_{\delta,j}$. This agglomerative clustering procedure is stopped when $d > \delta$, which results in a number of sets $s_{\delta,i}$. Each set is transformed into a group $g_{\delta,i}$ that holds all alerts of set $s_{\delta,i}$ as a sequence sorted by their occurrence time stamps as in Eq. 7.6.

$$g_{\delta,i} = \{(a_1, a_2, \ldots, a_n), \forall a_i \in s_{\delta,i} : a_1.t \leq a_2.t \leq \cdots \leq a_n.t\} \tag{7.6}$$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

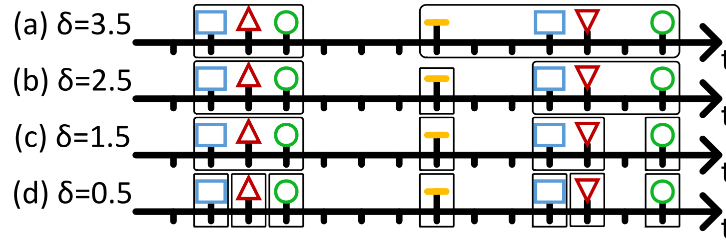


Figure 7.3: Alert occurrences duplicated over four parallel timelines show the formation of alert groups based on alert interval times. Larger intervals (top) allow more elapsed time between alerts and thus lead to fewer and larger groups compared to smaller intervals (bottom).

Equation 7.7 defines the set of all groups for a specific δ as their union.

$$\mathcal{G}_{\delta} = \bigcup_{i \in \mathbb{N}} g_{\delta,i} \tag{7.7}$$

This group formation strategy is exemplarily visualized in Fig. 7.3. The figure shows alert occurrences of types $\{\square, \triangle, \circ, -\}$ in specific patterns duplicated over four timelines with different δ. The sequence $(\square, \triangle, \circ)$ at the beginning of the timelines occurs with short alert interval times and that a similar sequence $(\square, \triangledown, \circ)$ occurs at the end, but involves $\triangledown$ instead of its variant $\triangle$ and has an increased interval time between $\triangledown$ and $\circ$. Nevertheless, due to the similar compositions of these two alert sequences, it is reasonable to assume that they are two manifestations of the same root cause.

In this example, each tick in the figure marks a time span of 1 unit. In timeline (d), all alerts end up in separate groups, because no two alerts yield a sufficiently small interval time lower than $\delta = 0.5$, i.e., $\mathcal{G}_{0.5} = \{(\square), (\triangle), (\circ), (-), (\square), (\triangledown), (\circ)\}$. In timeline (c) where alerts are grouped using $\delta = 1.5$, two groups that contain more than a single alert are formed, because the grouped alerts occur within sufficiently close temporal proximity, i.e., $\mathcal{G}_{1.5} = \{(\square, \triangle, \circ), (-), (\square, \triangledown), (\circ)\}$. Considering the results for $\mathcal{G}_{2.5} = \{(\square, \triangle, \circ), (-), (\square, \triangledown, \circ)\}$ in timeline (b) shows that the aforementioned repeating pattern $(\square, \triangle, \circ)$ and its variant end up in two distinct groups. This is the optimal case, since subsequent steps for group analysis could determine that both groups are similar and thus merge them into a meta-alert as shown in Sect. 7.1.1. A larger value for delta, e.g., $\delta = 3.5$ that yields $\mathcal{G}_{3.5} = \{(\square, \triangle, \circ), (-, \square, \triangledown, \circ)\}$ in timeline (a), adds alert of type $-$ to form group $(-, \square, \triangledown, \circ)$, which is not desirable since this decreases its similarity to group $(\square, \triangle, \circ)$. This example thus shows the importance for an appropriate selection of the interval threshold for subsequent analyses.

Note that this strategy for temporal grouping has several advantages over sliding time-windows. First, instead of time window size and step width, only a single parameter that specifies the maximum delta time between alerts is required, which reduces complexity of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

parameter selection. Second, it ensures that alerts with close temporal proximity remain in the same group given any delta larger than their interval times, while intervals of sliding time-windows possibly break up groups by chance. Third, related sequences with variable delays are more likely to end up in the same group, e.g., two groups with similar alerts but varying delays are found for $\delta = 2.5$ in Fig. 7.3.

Despite these benefits, pure time-based grouping suffers from some drawbacks compared to knowledge-based clustering methods, e.g., grouping by classification messages. As seen in the example from Fig. 7.3, the quality of the resulting grouping is highly dependent on a selection of the parameter δ that fits the typical time interval of the events to be grouped. Another issue is that randomly occurring alerts, e.g., false positives, are incorrectly allocated to groups if they occur in close proximity to one of the grouped alerts, and may even connect two or more groups into a single large group if they happen to occur in between and in sufficiently high amount or close proximity to both groups. As we will outline in the following sections, our approach mitigates these problems by finding groups using several values for δ in parallel.

Similarity Computation

Other than clustering based on predefined alert types, time-based grouping only acts as a preparatory step for subsequent analyses. In particular, a similarity measure for alert groups is required that allows to determine which groups of alerts are likely generated from the same root cause. Only then it is possible to cluster groups by their similarities and in turn generate meta-alerts by merging alert groups that end up in the same clusters. We therefore define function *group_sim* in Eq. 7.8 that computes the similarity of any two groups $g, h \in \mathcal{G}_\delta$.

$$group_sim : g, h \in \mathcal{G}_\delta \rightarrow [0, 1] \tag{7.8}$$

Analogous to alert similarity computation (cf. Sect. 7.1.2), the similarity between any non-empty group $g \in \mathcal{G}_\delta$ and itself is 1 and the similarity to an empty object is 0. However, we do not impose symmetry on the function, since it can be of interest to measure whether one group is contained in another possibly more abstract group, such as a meta-alert. Details on such a similarity function are discussed in Sect. 7.3.2. In the following section, we first explain the representation of meta-alerts and then introduce matching strategies for similarity computations between groups.

7.1.4 Meta-Alerts

We generate meta-alerts by merging groups, which relies on merging alerts within these groups. In the following, we first introduce features that support the representation of merged alerts and then outline group merging strategies for similarity computations and meta-alert generation.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Alert Merges

As outlined in Sect. 7.1.2, alerts are semi-structured objects, i.e., data structures that contain key-value pairs, and are suitable for similarity computation. However, aggregating similar alerts into a merged object that is representative for all allocated alerts is non-trivial, because single alert objects may have different keys or values that need to be taken into account.

For example, the failed login alert in Fig. 7.2 contains the attribute "`srcuser`" with value "`daryl`" in the "`data`" object. Since a large number of users may trigger such alerts, this event type occurs with many different values for attribute "`srcuser`" over time. An aggregated alert optimally abstracts over such attributes to represent a general failed login alert that does not contain any user information. The computed similarity between such an aggregated alert and any specific alert instance is independent of attributes that are known to vary, i.e., only the presence of the attribute "`srcuser`" contributes to similarity computations, but not its value. Note that this assumes that keys across alerts have the same semantic meaning or that keys with different names are correctly mapped if alert formats are inconsistent, e.g., keys "`src_user`" and "`srcuser`".

We incorporate merging of alerts by introducing two new types of values. First, a *wildcard* value type that indicates that the specific value of the corresponding key is not expressive for that type of alert, i.e., any value of that field will yield a perfect match just like two coinciding values. Typical candidates for values replaced by wildcards are user names, domain names, IP addresses, counts, and timestamps. Second, a *mergelist* value type that comprises a finite set of values observed in several alerts that are all regarded as valid values, i.e., a single matching value from the mergelist is sufficient to yield a perfect match for this attribute present in two compared alerts. The mergelist type is useful for discrete values that occur in variations, e.g., commands or parameters derived from events. Deciding whether an attribute should be represented as a wildcard or mergelist is therefore based on the total number of unique values observed for that attribute (see Sect. 7.3.3).

We define that each attribute key $k \in \kappa_a$ of an aggregated alert a that is the result of a merge of alerts $A \subseteq \mathcal{A}$ is represented as either a wildcard or mergelist as in Eq. 7.9.

$$a.k = \begin{cases} \mathit{wildcard}\,() \\ \mathit{mergelist}\,(\bigcup_{b \in A} b.k) \end{cases} \tag{7.9}$$

Note that Eq. 7.9 also applies for nested keys, i.e., values within nested objects stored in the alerts. Since our approach is independent of any domain-specific reasoning, a manual selection of attributes for the replacement with wildcards and mergelists is infeasible. The function *alert_merge* thus automatically counts the number of unique values for each attribute from alerts $A \subseteq \mathcal{A}$ passed as a parameter, selects and replaces them with the appropriate representations, and returns a new alert object a that represents a merged alert that is added to all alerts $\mathcal{A}$ as shown in Eq. 7.10-7.11.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$$a = alert_merge(A), \quad A \subseteq \mathcal{A} \tag{7.10}$$
$$\mathcal{A} \Leftarrow a \tag{7.11}$$

Note that we use the operation $\Leftarrow$ to indicate set extensions, i.e., $\mathcal{A} \Leftarrow a \iff \mathcal{A}' = \mathcal{A} \cup \{a\}$. We drop the prime of sets like $\mathcal{A}'$ in the following for simplicity and assume that after extension only the new sets will be used. The extension of $\mathcal{A}$ implies that merged alerts are also suitable for similarity computation and merging with other alerts or merged alerts. We will elaborate on the details of the alert merging procedure in Sect. 7.3.3. The next section will outline the role of alert merging when groups are merged for meta-alert generation.

Group Merges

Similar to merging of alerts that was discussed in the previous section, a merged group should represent a condensed abstraction of all groups used for its generation. Since each group should ideally comprise a similar sequence of alerts, it may be tempting to merge groups by forming a sequence of merged alerts, where the first alert is merged from the first alerts in all groups, the second alert is merged from the second alerts in all groups, and so on. Unfortunately, this is infeasible in practice, because alert sequences are not necessarily ordered, involve optional alerts, or are affected by false positives causing that alert positions in sequences are shifted. To alleviate this issue, it is necessary to find matches between the alerts of all groups to be merged. In the following, we describe three matching strategies used in our approach that are suitable for group similarity computation as well as meta-alert generation.

Exact matching. This strategy finds for each alert in one group the most similar alert in another group and uses these pairs to determine which alerts to merge. The idea of finding these matches is depicted in the left side of Fig. 7.4, where lines across groups g_1, g_2, g_3 indicate which alerts were identified as the most similar. As expected, alerts of the same type are matched, because they share several common attributes and values that are specific to their respective types. The figure also shows that correct alerts are matched even though the second and third alert in g_2 are in a different order than in g_1 and g_3. In addition, note that the alert of type $\triangledown$ in g_1 is correctly matched to the related alert type $\triangle$ in g_2 and that the merged group thus contains the merged alert type ⧖ at that position. In addition, there is a missing alert of type $\circ$ in g_3 that leads to an incomplete match. Nevertheless, the alert of type $\circ$ ends up in the merged group, because it occurs in the majority of all merged groups and is therefore considered to be representative for this root cause manifestation.

When only two groups are considered, this matching method is also suitable for measuring their similarity. In particular, this is achieved by computing the average similarity of all matched alerts, where non-matching alerts count as total mismatches. The similarity score is further enhanced by incorporating an edit distance [Nav01] that measures the

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

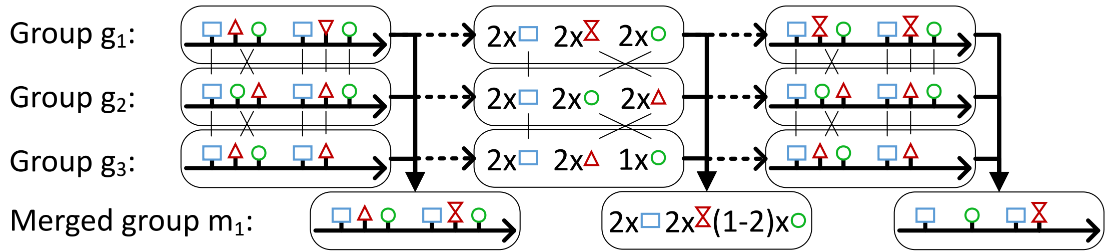


Figure 7.4: Merging strategies for alert groups. Left: Finding exact matches between alert pairs. Center: Matching representatives using a bag-of-alerts model. Right: Matching using alert sequence alignment.

amount of inserts, removes, and substitutions of alerts, i.e., misalignments such as the occurrence of $(\circ, \triangle)$ instead of $(\triangle, \circ)$ in g_2.

While the exact matching strategy yields accurate group similarities, it is rather inefficient for large groups. The reason for this is that computing the pairwise similarities of all alerts requires quadratic runtime with respect to group sizes. We therefore only use this strategy when the number of required comparisons for groups g, h does not exceed a limit $l_{bag} \in \mathbb{N}$, i.e., $|g| \cdot |h| \leq l_{bag}$, where $|g|$ denotes the size of group g. In the following, we outline an alternative strategy for larger groups.

Bag-of-alerts matching. For this strategy, we transform the alert sequences of all groups into a bag-of-alerts model following the well-known bag-of-words model [MSR08]. This is accomplished by incrementally clustering the alerts within each group using a certain similarity threshold $\theta_{alert} \in [0, 1]$, where each alert a that is sufficiently similar to one of the initially empty set of cluster representatives R, i.e., $alert_sim(r, a) \geq \theta_{alert}, \forall r \in R$, is added to the list C_r that stores all alerts of that cluster, i.e., $C_r \Leftarrow a$, or forms a new cluster with itself as a representative otherwise, i.e., $R \Leftarrow a$. Once all alerts of a group are processed, the bag-of-alerts model for that group is generated by merging all alerts in each cluster, i.e., $alert_merge(C_r), \forall r \in R$.

The matching procedure then finds the pairs of these merged alerts that yield the highest similarities across groups and aggregates them by identifying lower and upper limits of their corresponding cluster sizes $|C_r|$ in each group. The advantage in comparison to the exact matching strategy is that the number of necessary similarity computations is reduced to the product of the number of clusters per group, which is controllable through θ_{alert}. Note that the speedup stems from the fact that the computation of the bag-of-alerts model only has to be carried out once for each group, but then enables fast matching with all other groups.

The center part of Fig. 7.4 shows bag-of-alert models for sample groups, where alerts of types $\triangle$ and $\triangledown$ in g_1 are merged to ⧖, which is then matched to $\triangle$ in g_2 and g_3 before they are once again merged for the generation of the meta-alert. Since alert type $\circ$ occurs twice in g_1 and g_2, but only once in g_3, the meta-alert uses a range with minimum limit

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$l_{min} = 1$ and maximum limit $l_{max} = 2$ to describe the occurrence frequency of this alert type.

This strategy also supports measuring the similarity of two groups g, h by averaging the relative differences of occurrence counts, which yields the highest possible similarity of 1 (i.e., a perfect match) if the respective counts coincide or their intervals overlap, and $min(l_{max,g}, l_{max,h})/max(l_{min,g}, l_{min,h})$ otherwise. Alerts without a match are considered as total mismatches and contribute the lowest possible similarity score of 0 to the average. We favored this similarity metric over existing measures such as cosine similarity [MSR08], because it allows a more intuitive representation of lower and upper occurrence limits which supports human interpretation of meta-alerts.

The downside of the bag-of-alerts strategy is that information on the order of the alerts is lost. However, it is possible to resolve this issue by combining the original alert sequence with the bag-of-alerts model. In the following, we outline this addition to the bag-of-alerts matching.

Alignment-based matching. To incorporate alignment information for large clusters that are not suited for the exact matching strategy, it is necessary to store the original sequence position of all clustered alerts during generation of the bag-of-alerts model of each group. This information enables to generate a sequence of cluster representatives. For example, the right side of Fig. 7.4 shows that group g_1 has sequence $(\square, ⧖, \circ, \square, ⧖, \circ)$, because the occurrences of $\triangle$ and $\triangledown$ have been replaced by their cluster representative ⧖ that was generated in the bag-of-alerts model. Note that this strategy is much faster for large groups than the exact matching strategy, because it enables to reuse the matching information of representative alerts from the bag-of-alerts model instead of finding matches between all alerts. Since the corresponding sequence elements across groups are known, it is simple to use sequence alignment algorithms for merging and similarity computation.

We decided to merge the sequences using longest common sequence (LCS) [Nav01], because it enables to retrieve the common alert pattern present in all groups and thereby omit randomly occurring false positive alerts [LSW+19]. The example in Fig. 7.4 shows that this results in a sequence of representatives $(\square, \circ, \square, ⧖)$ that occurs in the same order in all groups. Using the LCS also enables to compute the sequence similarity of two groups g, h by $|LCS(g, h)| / min(|g|, |h|)$, which we use to improve the bag-of-alerts similarity by incorporating it as a weighted term after averaging.

Equation 7.12 defines a function that takes a set of groups $G \subseteq \mathcal{G}_\delta$ and automatically performs all aforementioned merging strategies to generate a new group g.

$$g = group_merge(G), \quad G \subseteq \mathcal{G}_\delta \tag{7.12}$$

$$\mathcal{G}_\delta \Leftarrow g \tag{7.13}$$

Analogous to merges of single alerts, Eq. 7.13 indicates that merges of alert groups have the same properties as normal groups and therefore support similarity computation and

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

merging. In the previous sections, we defined several functions required for meta-alert generation. The following section will embed all aforementioned concepts in an overall procedure.

7.2 Framework

This section outlines a procedure for meta-alert generation based on the aforementioned concepts and functions. We first describe the overall approach and then present its steps in two scenarios.

7.2.1 Overview

Our procedure reads in a sequence of alerts from one or multiple IDSs. The first step is to form groups from these incoming alerts as outlined in Sect. 7.1.3. Unfortunately, manually specifying δ as the maximum allowed interval time between alerts is non-trivial, because it requires a high amount of knowledge about alert interactions and expected attack pattern structures. Even worse, different alert patterns may require specific settings for δ that are incompatible with each other. To resolve this issue, we carry out group formation in parallel for several values $\delta \in \Delta$ similar to Fig. 7.3, where Δ is the set of all values for δ. This increases the chance that valid and usable meta-alerts are found for various types of attacks. In addition, it forms a hierarchical structure of alert patterns, where small δ values generate groups that contain mainly technically linked alerts, e.g., a failed login alert that occurs simultaneously with a frequency alert for such events, and groups generated by large δ values that contain sequentially executed attack steps [LSW+19]. For simplicity, we only use δ in the following and implicitly assume that all computations are carried out for all $\delta \in \Delta$ analogously.

We define a set of meta-alerts $\mathcal{M}_\delta$ that holds merged groups. Note that the index δ indicates that meta-alerts are generated for all δ values separately, i.e., groups formed by different δ values are not merged together. The reason for this is that merging groups that were partially formed from the same alert occurrences may lead to overly generalized meta-alerts and thus loss of information. For example, consider the groups from Fig. 7.3, where group $(\Box, \triangledown, \circ) \in \mathcal{G}_{2.5}$ and group $(-, \Box, \triangledown, \circ) \in \mathcal{G}_{3.5}$ contain three identical alerts and may thus be considered similar enough for merging. This is not desirable, since the resulting merge will involve the alert type $-$, which is not part of the actual attack pattern $(\Box, ⧗, \circ)$ that is the result of merging $(\Box, \triangle, \circ) \in \mathcal{G}_{2.5}$ and $(\Box, \triangledown, \circ) \in \mathcal{G}_{2.5}$. Such cascading merges occurring over different δ values could mostly be prevented by prohibiting merges of groups that contain identical alert instances. However, to avoid this issue altogether and to enable a thorough evaluation for each δ value, we process all meta-alerts sets $\mathcal{M}_\delta$ isolated.

A new group $g \in \mathcal{G}_\delta$ is incrementally added to the set of meta-alerts $\mathcal{M}_\delta$ by finding the meta-alert $m \in \mathcal{M}_\delta$ with the highest similarity, i.e., $sim = max_{m \in \mathcal{M}_\delta}(group_sim(g, m))$. If the similarity is higher than a predefined threshold $\theta_{group} \in [0, 1]$, i.e., $sim \geq \theta_{group}$,

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the group is added to the most similar meta-alert m, otherwise a new meta-alert is generated for this group.

While it may be typical for incremental approaches, it is not recommended to merge group g and meta-alert m directly, i.e., $m = group_merge(\{g, m\})$, because this causes that meta-alerts over-generalize over time. The reason for this is that a single incorrect allocation of a group to a meta-alert extends mergelists of attributes or introduces wildcards, which will increase the similarity of the meta-alert to all other groups and thus make it more susceptible to incorrect allocations in a self-enforcing loop. As a solution, we store allocated groups for each meta-alert in a so-called *knowledge base* $\mathcal{K}_\delta$, where $K_m \subseteq \mathcal{K}_\delta$ is the set of all groups allocated to meta-alert m. For group g and meta-alert m where $sim \geq \theta_{group}$, we therefore update the knowledge base $K_m \Leftarrow g$ and regenerate meta-alert $m = group_merge(K_m)$ from all groups. The advantage of this strategy is that it allows to generate meta-alerts from more than two groups at the same time, which is more robust against single group misallocations since attribute merging can be based on majority decision or predefined minimum occurrences. In addition, it allows to adapt group allocations in the knowledge base, e.g., reallocate individual groups that turn out to be incorrectly classified without the need to remove the meta-alert, split one meta-alert m into multiple meta-alerts by extracting subsets of K_m, or merge meta-alerts by unifying their groups.

Storing all identified groups in the knowledge base is usually infeasible in practice due to limited available memory as well as increasing runtime for merging larger amounts of groups. We therefore use a queue to enable the following strategies for storing groups in each $K_m \subseteq \mathcal{K}_\delta$:

- **Unlimited storage.** This strategy implies that queue sizes grow indefinitely. Such a strategy is useful for forensic analyses, where the total number of groups is limited and known to be sufficiently small, and it is thus possible to store all groups.

- **Linear storage.** With this strategy, the size of the queues is limited. Once the queue is full, adding a new group will cause the oldest group in the queue to be removed.

- **Logarithmic storage.** First, the queue is filled to its maximum size. Then, any newly added group will replace the last group with probability 1/2, move the last group one position lower with probability 1/4, move each of the last two groups one position lower with probability 1/8, etc. This ensures that groups at the beginning of the queue remain in the queue for a longer time span and that the groups stored in the queue collectively represent a more diverse set. This strategy is therefore especially useful when related alerts are expected to occur over long time intervals, e.g., when they are collected from different environments.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

In the next section, we show the individual steps of the procedure by two application cases. For simplicity, we assume that the unlimited storage strategy is used and thus treat each K_m as a set.

7.2.2 Scenarios

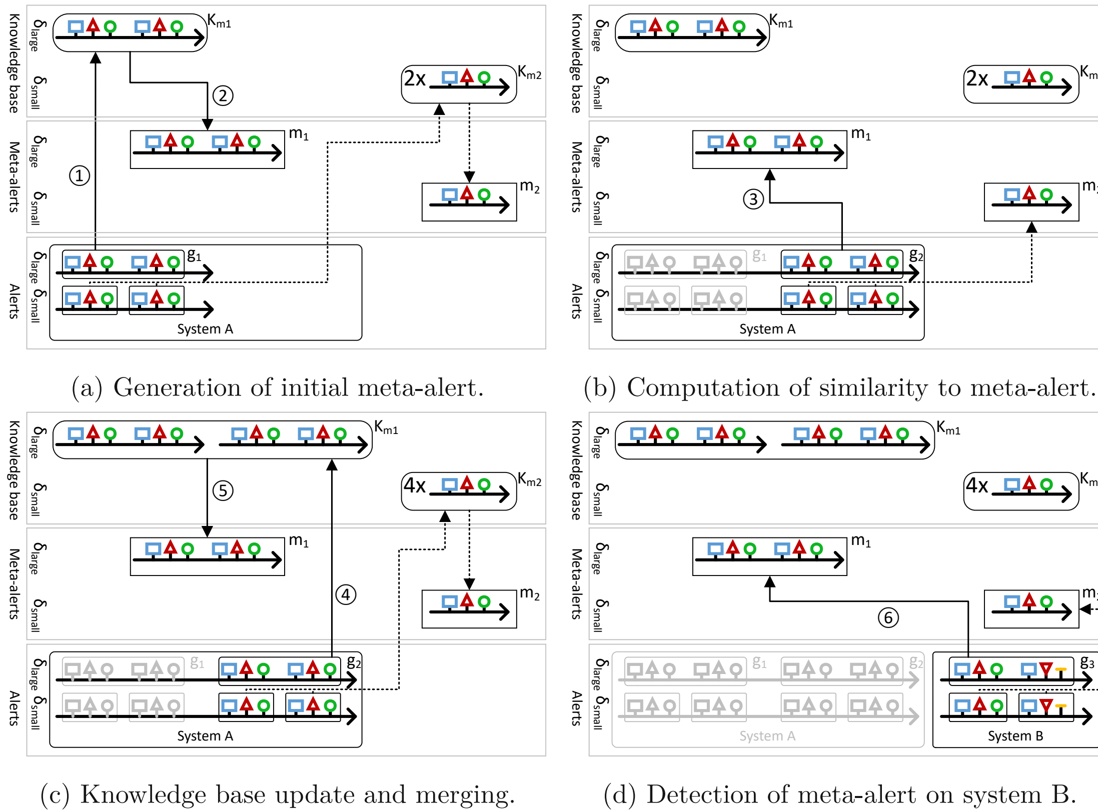
We select two scenarios to explain the approach for meta-alert generation in the following. The first scenario is displayed in Figs. 7.5a-7.5d and deals with reusing meta-alerts for classification of alerts occurring on other systems. Thereby, each of the figures depicts the state of the incremental alert aggregation framework at a specific point in time. Moreover, we constructed the figures to show the alert occurrences $\mathcal{A}$ and the formed groups $\mathcal{G}_\delta$ in the bottom, the generated meta-alerts $\mathcal{M}_\delta$ in the center, and the knowledge base $\mathcal{K}_\delta$ in the top. Note that in each of these blocks we display two sections, one for a δ_{large} value (top) and one for a δ_{small} value (bottom), where $\delta_{large} > \delta_{small}$. For simplicity, we focus only on groups generated by the δ_{large} value in the first scenario.

Figure 7.5a depicts the state of the framework after one group $g_1 = (\square, \triangle, \circ, \square, \triangle, \circ)$ was formed for δ_{large}, i.e., the time passed after the last alert occurrence exceeds δ_{large}. Since no meta-alerts exist at this point, a new meta-alert m_1 is created by instantiating group $K_{m_1} \Leftarrow g_1$ so that $K_{m_1} = \{g_1\}$ in the knowledge-base as indicated by step (1), and generating meta-alert $m_1 = group_merge(K_{m_1})$ as indicated by step (2). Note that meta-alert m_1 involves the same alert sequence with identical attributes as group g_1, but all values are represented as mergelists as outlined in Sect. 7.1.4.

Figure 7.5b depicts the occurrence of another group $g_2 = (\square, \triangle, \circ, \square, \triangle, \circ)$ on system A. Step (3) shows that the similarity between g_2 and each $m \in \mathcal{M}_\delta$ is computed, in particular, only the similarity $sim_{g_2} = group_sim(g_2, m_1)$ is computed since only $m_1 \in \mathcal{M}_\delta$ exists. Due to the fact that both groups g_1, g_2 involve the same alert sequence, we assume that their similarity exceeds a predefined threshold θ_{group}, i.e., $sim_{g_2} = group_sim(g_2, m_1) \geq \theta_{group}$, indicating that g_1 relates to the same root cause as m_1 and should therefore be aggregated. Figure 7.5c shows that this is achieved by adding group g_2 to the knowledge base storing the groups allocated to m_1, i.e., $K_{m_1} \Leftarrow g_2$ so that $K_{m_1} = \{g_1, g_2\}$, as indicated by step (4). Adding a group to K_{m_1} triggers a regeneration of meta-alert m_1 as indicated by step (5), i.e., $m_1 = group_merge(K_{m_1})$. Assuming that all alerts in groups g_1, g_2 have the same attributes and values, the resulting meta-alert m_1 remains unchanged.

Figure 7.5d displays group $g_3 = (\square, \triangle, \circ, \square, \triangledown, -)$ occurring in system B at some point after m_1 is generated from alerts on system A. Step (6) depicts the similarity computation $sim_{g_3} = group_sim(g_3, m_1)$. Note that only the first four out of six alerts in m_1 and g_3 are identical, while the fifth alert $\triangledown$ of g_3 is a variation of alert $\triangle$ in m_1 and the sixth alert is of a different type. If the similarity is sufficiently high, i.e., $sim_{g_3} \geq \theta_{group}$, the occurrence of the group is interpreted as a detection of the attack represented by m_1. Otherwise, the group is assumed to depict a new unknown attack, causing that a new meta-alert is generated from group g_3 similar to steps (1)-(2).

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub



(a) Generation of initial meta-alert.

(b) Computation of similarity to meta-alert.

(c) Knowledge base update and merging.

(d) Detection of meta-alert on system B.

Figure 7.5: Scenario for cross-system alert recognition. Steps (1)-(5) show the meta-alert generation procedure using alerts from system A. Steps (6) indicates the detection of a similar alert group on system B.

The procedure for groups identified for δ_{small} is indicated by dashed arrows and works analogously. Figures 7.5a-7.5c show that four groups occurring on system A are iteratively added to the knowledge base K_{m_2} and are merged to a single meta-alert $m_2 = (\square, \triangle, \circ)$. Figure 7.5d shows that two groups are identified for δ_{small}, of which one comprises the same alert sequence as meta-alert m_2 and is thus similar enough to yield a successful detection of the same attack pattern, while the other group is rather dissimilar and could therefore lead to the generation of a new meta-alert.

The second scenario is visualized in Figs. 7.6a-7.6d and focuses on merging alert groups across systems. For simplicity, the following description focuses on groups generated by the δ_{small} value. Similar to the first scenario, steps (1) and (2) in Fig. 7.6a indicate the generation of meta-alert m_1 from the first group $g_1 = (\circ, \triangle, \circ, \triangle)$ on system A, so that $K_{m_1} = \{g_1\}$. As shown in Fig. 7.6b, the difference to the first scenario is that group $g_2 = (\circ, \triangledown, \circ, \triangleright)$ in system A has variations of alert type $\triangle$ occurring in the second and fourth alert. For the sake of example, we consider alert types $\{\triangle, \triangledown, \triangleright, \triangleleft\}$ to be similar alerts with the same set of attributes but different values in one specific

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

attribute, e.g., a different user name (cf. Sect. 7.1.4). Despite these variations, group g_2 involves similar alert types and therefore yields a sufficiently high similarity, i.e., $sim_{g_2} = group_sim(g_2, m_1) \geq \theta_{group}$. As a consequence, group g_2 added to the knowledge base of meta-alert m_1 in step (3), i.e., $K_{m_1} \Leftarrow g_2$ so that $K_{m_1} = \{g_1, g_2\}$, which is in turn used to update meta-alert $m_1 = group_merge(K_{m_1})$ as indicated by step (4). Since the resulting meta-alert m_1 is a merge of groups g_1, g_2, its second alert is a merge of alert types $\{\triangle, \triangledown\}$ and its fourth alert is a merge of alert types $\{\triangle, \triangleright\}$.

Different to the first scenario, alert groups from system B are used to generate a cross-system meta-alert. Figure 7.6c shows group $g_3 = (\circ, \triangle, \circ, \triangle)$, which involves alerts $\triangle$ on the second and fourth positions. Since $\triangle$ is part of the aggregated alerts of meta-alert m_1, similarity $sim_{g_3} = group_sim(g_3, m_1) \geq \theta_{group}$ is high and the group is thus added to K_{m_1}. While also the second alert $\triangledown$ of group $g_4 = (\circ, \triangledown, \circ, \triangleleft)$ yields a perfect match with the second alert ⧖ of meta-alert m_1, the fourth alert $\triangleleft$ of group g_4 is not part of m_1 and thus slightly decreases similarity $sim_{g_4} = group_sim(g_4, m_1)$, which is nonetheless assumed to exceed θ_{group} since all other alerts match. Therefore, $K_{m_1} \Leftarrow g_4$ so that $K_{m_1} = \{g_1, g_2, g_3, g_4\}$ as indicated by step (5). Note that in all four groups, the second alert is one of $\{\triangle, \triangledown\}$, and the fourth alert is one of $\{\triangle, \triangleright, \triangleleft\}$. When generating m_1 after updating K_{m_1} in step (6), the affected attribute of the fourth alert is therefore replaced with a wildcard so that $m_1 = (\circ, ⧖, \circ, *)$. Since the wildcard matches all values, both groups g_5, g_6 displayed in Fig. 7.6d yield perfect matches with m_1, even though alert $\triangledown$ in g_6 does not occur in any group of K_{m_1}.

Inspecting meta-alert m_2 in Fig. 7.6d that was generated by groups of system A and system B using δ_{large} shows that the sequence of merged alerts differs from m_1, e.g., alert types $\triangle$ and $\triangledown$ occur instead of alert type ⧖. Since this scenario depicts just an exemplary demonstration that is not based on real alerts, it is not possible to determine which of the meta-alerts m_1, m_2 is better suited for detection. However, both scenarios suggest that it is reasonable to consider multiple values for δ to generate several different meta-alerts that cover a large variety of attack manifestations.

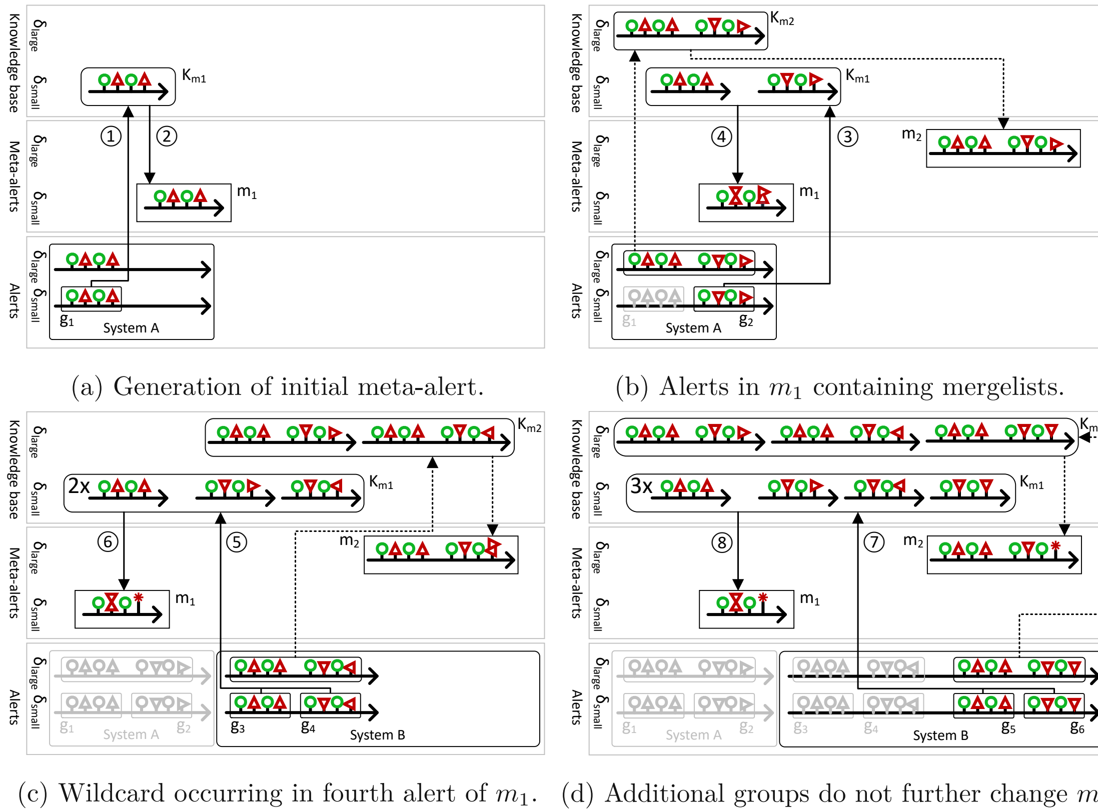
7.3 Implementation of the Framework

The previous sections provided a theoretical overview of alerts, alert groups, and meta-alerts. Thereby, we defined abstract functions for similarity computation and merging of these concepts to introduce a procedure for automatic meta-alert generation. In this section, we will go into more detail about these functions and discuss their properties with the aid of pseudo code.

7.3.1 Alert Similarity

This section outlines the alert similarity function from Sect. 7.1.2. Since alert objects contain nested dictionaries, recursions are used for similarity computation. Algorithm 1 shows the recursion start in Line 2 of procedure $alert_sim$ with parameters $a, b \in \mathcal{A}$. The

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



(a) Generation of initial meta-alert.

(b) Alerts in m_1 containing mergelists.

(c) Wildcard occurring in fourth alert of m_1.

(d) Additional groups do not further change m_1.

Figure 7.6: Scenario for cross-system alert merging. Steps (1)-(8) alternate between knowledge base extensions and repeated meta-alert generation.

recursion returns scores for matching and mismatching attributes, which are normalized to a single similarity (Line 3).

The recursive function is realized by iterating over all common attributes of alerts a, b (Line 7). For each of these attributes, the function adds up the achieved match and mismatch scores by comparing the types of the respective values to ensure suitable comparison. This includes (i) dictionary types (Lines 9-10) that call the recursive function with the nested dictionaries, (ii) wildcard (WC) types (Lines 11-12) that always count as matches, (iii) mergelist (ML) types (Lines 13-15) that only count as a mismatch if no two elements are the same in both mergelists and yield larger match scores for more identical elements present in both mergelists corresponding to the overlap metric [MSR08], (iv) standard list types (Lines 16-20) that measure the ratio of common elements with respect to all elements in both lists, and (v) standard value types (Lines 21-25) match if the values of the same attribute in both alerts are identical, and count as mismatches otherwise. We place the check of standard value types at the end to ensure comparison is not carried out on different data types. Finally, the number of keys that are present in one of alerts a, b but not the other contributes to the mismatch score (Line 29), where $\ominus$

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

Algorithm 1 Alert similarity computation.

```
1:  procedure ALERT_SIM(a, b)
2:      match, mismatch ← alert_sim_rec(a, b)
3:      return match/(match + mismatch)
4:  end procedure
5:  procedure ALERT_SIM_REC(a, b)
6:      match ← 0, mismatch ← 0
7:      for all k ∈ {κ_a ∩ κ_b} do
8:          mat ← 0, mis ← 0
9:          if type(a.k) = dict  and
                type(b.k) = dict then
10:             mat, mis ← alert_sim_rec(a.k, b.k)
11:         else if type(a.k) = WC  or
                type(b.k) = WC then
12:             mat ← 1
13:         else if type(a.k) = ML  or
                type(b.k) = ML then
14:             mat ← |a.k ∩ b.k| / min(|a.k| , |b.k|)
15:             mis ← (mat > 0) ? 1 : 0
16:         else if type(a.k) = list  or
                type(b.k) = list then
17:             com ← |a.k ∩ b.k|
18:             dif ←max(|(a.k \ b.k| , |b.k \ a.k|)
19:             mat ← com/(com + dif)
20:             mis ← dif/(com + dif)
21:         else if a.k = b.k then
22:             mat ← 1
23:         else
24:             mis ← 1
25:         end if
26:         match ← match + mat
27:         mismatch ← mismatch + mis
28:     end for
29:     mismatch ← mismatch + |κ_a ⊖ κ_b|
30:     return match, mismatch
31: end procedure
```

Algorithm 2 Group similarity computation.

```
1:  procedure GET_MATCHING(g, h)
2:      pairs ← list(), sims ← list()
3:      for all a ∈ g do
4:          for all b ∈ h do
5:              pairs.append((a, b))
6:              sims.append(alert_sim(a, b))
7:          end for
8:      end for
9:      sort(pairs, by=sims, order=descending)
10:     return pairs
11: end procedure
12: procedure GROUP_SIM(g, h)
13:     sim ← 0
14:     used_a ← ∅
15:     used_b ← ∅
16:     for all a, b ∈ get_matching(g, h) do
17:         if a ∉ used_a  and b ∉ used_b then
18:             used_a ⇐ a
19:             used_b ⇐ b
20:             sim ← sim + alert_sim(a, b)
21:         end if
22:     end for
23:     return sim/max(|g|, |h|)
24: end procedure
```

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

is the operator for symmetric difference.

The function fulfills all requirements specified in Sect. 7.1.2. The normalization in Line 3 ensures that the resulting similarity scores lie within the interval $[0, 1]$, where 1 indicates that all keys and values match, and 0 indicates that none of the keys and values match. Moreover, the similarity function is symmetric, since all operations on alerts a, b are symmetric. Due to the incorporation of wildcard and mergelist types, the similarity function implicitly measures how well one alert is represented by another merged alert when comparing alerts on different levels of abstraction.

The pseudo code contains the most important type comparisons, however, the presented checks are not exhaustive. For example, match and mismatch scores of specific type combinations such as list and mergelist are neglected for brevity. Furthermore, the pseudo code does not incorporate weights, which allow to steer the contribution of attributes to the similarity, e.g., attribute "timestamp" should have weight 0, because the occurrence times of alerts are not expected to match and should not prevent perfect matches. We refer to our implementation for more details.

7.3.2 Group Similarity

This section defines a similarity measure for groups that fulfills the requirements defined in Sect. 7.1.3. In addition, we present the pseudo code for the comparison method that we refer to as the exact matching strategy in Sect. 7.1.4. We select this strategy, because it establishes the basis for the other matching strategies that involve additional steps such as clustering of alerts within groups.

Algorithm 2 shows the pseudo code for the matching procedure *get_matching* as well as the group similarity function *group_sim*. The matching procedure takes two groups $g, h \in \mathcal{G}_\delta$ as parameters and computes the pairwise similarities between the alerts of each group (Lines 3-8) and stores both the alert pairs and their similarities in respective lists. Then, the pairs are sorted in decreasing order so that the most similar pairs of alerts are at the beginning of the list (Line 9).

As mentioned in Sect. 7.1.4, computing the alert similarity between all alerts may cause a loss of performance when large groups are compared. The bag-of-alerts matching strategy solves this issue by calling the function with representative alerts for each group, which largely reduces the number of required alert similarity computations. More details on the similarity computation in the bag-of-alerts model is stated in Sect. 7.1.4 and provided in our implementation.

The function *group_sim* also takes two groups $g, h \in \mathcal{G}_\delta$ as parameters and computes an aggregated similarity of all contained alerts. For this, it first finds the matching between the alerts of the groups and then iterates over all returned pairs (Line 16). Lines 17-19 ensure that each alert is only considered at most once, i.e., alert pairs where one of the alerts was already used for similarity computation are skipped. Line 20 shows that the aggregated group similarity is the sum of all individual pair similarities. Finally, the

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

resulting similarity is normalized in Line 23 to lie in the interval $[0, 1]$. This line also ensures that alerts without matches due to different group sizes decrease the overall group similarity score. Moreover, the function yields a similarity of 1 if groups g, h are identical and a similarity of 0 when all alert pairs achieve a similarity of 0 as required in Sect. 7.1.3. Also note that the function is symmetric, but can easily be adapted to measure how well one group g fits to another group h, in particular, by replacing the division in Line 23 with $sim/|g|$.

Note that we neglected alignments [Nav01] in the pseudo code for brevity. This could be achieved by counting mismatching alerts as well as misalignments of matched alerts. We refer to our implementation, where we enhance the final similarity by incorporating such an alignment score.

7.3.3 Alert Merging

In this section, we outline a function for alert merging as specified in Sect. 7.1.4. The function generates a new alert that comprises wildcards and mergelists, which are represented by the two classes WC and ML that have already been used for alert similarity computation in Alg. 1.

Algorithm 3 shows the pseudo code for generating a merged alert. Parameters are a set of alerts $A \subseteq \mathcal{A}$ to be merged, a ratio $k_{min} \in [0, 1]$ specifying the minimal relative occurrence frequency of an attribute to be included in the merged alert, a ratio $v_{min} \in [0, 1]$ specifying the minimal relative occurrence frequency of a value to be included in an attribute of the merged alert, and a number $v_{max} \in \mathbb{N}_0$ specifying the maximum amount of values before mergelists are replaced by wildcards.

The procedure first extracts list $keys$ that holds all attributes present in the alerts, where identical keys are stored multiple times (Lines 2-5). Line 6 computes the occurrence frequencies of these keys using a count function, which yields dictionary $keys_count$ that holds all keys and their respective frequencies. Lines 6-11 then remove all keys with relative occurrence frequencies smaller than k_{min}. This step ensures that rare keys that do not occur in sufficiently many alerts of A are omitted.

An initially empty object c for the alert merge is defined in Line 12. Every remaining attribute key k is then added iteratively to that object in Lines 13-35. Thereby, the values of each key from all alerts are stored in a list (Lines 14-19). Note that a value may also be a list or mergelist, in which case the list $vals$ is extended with all values from that list. In case that all values of a particular key are dictionaries, the function $alert_merge$ is called recursively for that attribute (Lines 20-21). Otherwise, the values in the list are counted to remove all values that occur with relative frequencies lower than v_{min} (Lines 23-28). In case that no values remain, the number of values exceeds v_{max}, or one of the values is a wildcard, an attribute with key k and a wildcard as value is added to the merged alert c (Lines 29-30). Otherwise, all disclosed values are added to an attribute holding a mergelist (Line 32). Line 36 returns the merged alert c after processing all keys.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm 3 Alert merge computation.

```
1: procedure ALERT_MERGE(A, k_min, v_min, v_max)
2:     keys ← list()
3:     for all a ∈ A do
4:         keys.extend(κ_a)
5:     end for
6:     keys_count ← count(keys)
7:     for all k, freq ∈ keys_count do
8:         if freq/|A| < k_min then
9:             keys.remove(k)
10:        end if
11:    end for
12:    c ← dict()
13:    for all k ∈ keys do
14:        vals ← list()
15:        for all a ∈ A do
16:            if k ∈ κ_a then
17:                vals.append(a.k)
18:            end if
19:        end for
20:        if ∀type(v ∈ vals) = dict then
21:            c_k ← alert_merge(
                    vals, k_min, v_min, v_max)
22:        else
23:            vals_count ← count(vals)
24:            for all v, freq ∈ vals_count do
25:                if freq/|values| < v_min and
                        type(v) ≠ WC then
26:                    vals.remove(v)
27:                end if
28:            end for
29:            if |vals| = 0 or |vals| > v_max or
                    ∃type(v ∈ vals) = WC then
30:                c_k = WC()
31:            else
32:                c_k = ML(vals)
33:            end if
34:        end if
35:    end for
36:    return c
37: end procedure
```

Algorithm 4 Group merge computation.

```
1: procedure GROUP_MERGE(G, θ_alert, k_min, v_min,
        v_max)
2:     largest_group ← list()
3:     for all g ∈ G do
4:         if |g| > |largest_group| then
5:             largest_group ← g
6:         end if
7:     end for
8:     d ←dict()
9:     for all a ∈ largest_group do
10:        d[a] ← {a}
11:    end for
12:    for all g ∈ (G \ largest_group) do
13:        used_a ← ∅
14:        used_b ← ∅
15:        for all a, b ∈ get_matching(g,
                d.keys()) do
16:            if alert_sim(a, b) < θ_alert then
17:                break
18:            else if a ∉ used_a and
                    b ∉ used_b then
19:                used_a ⇐ a
20:                used_b ⇐ b
21:                d[b] ⇐ a
22:            end if
23:        end for
24:        for all missing ∈ (g \ used_a) do
25:            d[missing] ⇐ missing
26:        end for
27:    end for
28:    h ←list()
29:    for all A ∈ d.values() do
30:        c ← alert_merge(A, k_min, v_min, v_max)
31:        h.append(c)
32:    end for
33:    return h
34: end procedure
```


Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

As required in Sect. 7.1.4, every generated alert c is a possibly nested semi-structured object that only holds wildcards and mergelists in its attributes. Accordingly, it is possible to treat it like any other alert $a \in \mathcal{A}$, which includes similarity computations and merging.

7.3.4 Group Merging

This section discusses the group merging function introduced in Sect. 7.1.4 that is capable of generating meta-alerts, i.e., aggregated alert groups. The parameters of the function outlined in Alg. 4 are a set of groups $G \subseteq \mathcal{G}_\delta$, a similarity threshold for alerts $\theta_{alert} \in [0, 1]$, and the values $k_{min}, v_{min}, v_{max}$ required for the *alert_merge* function discussed in Sect. 7.3.3.

We introduced the bag-of-alerts model to alleviate performance issues that arise from determining alert matches between two large groups to compute their similarity. Unfortunately, group merging introduces a new problem, since not just two, but arbitrary numbers of groups can be merged at the same time. The main issue with that scenario is that finding alert matches between all pairs of groups is highly resource-intensive and should therefore be avoided. In the following, we solve this problem by merging groups incrementally, i.e., use one group as a representative that all other groups are merged to. This is also represented in Fig. 7.4, where alerts of both groups g_1, g_3 are matched with alerts of group g_2, but there is no alert matching taking place between groups g_1, g_2 themselves. In that scenario, group g_2 acts as the representative group.

Lines 2-7 in Alg. 4 show that we select the largest group in the set of groups G as the representative group, because it contains the most alerts and is thus the most likely to yield many alert matches with all other groups to be merged. We then define a dictionary d in Line 8 that holds lists of alerts to be merged in its values. For this, we first initialize it by adding all alerts of the largest group as keys and each alert in a list as their values. Lines 12-27 then append all matching alerts of other groups to these lists by iterating over all remaining groups. For each group, the alert matching is computed in Line 15. We iterate over all alert pairs ordered by their achieved similarity (cf. Alg. 2) and add the alerts to the best matching key of dictionary d. The iteration stops when the minimum similarity θ_{alert} is reached (Line 16). This check is necessary to avoid that alerts with low similarity are incorrectly merged with each other, resulting in over-generalized alerts in the merged group. In case that not all alerts of the currently processed group could be matched, e.g., if the achieved similarity to any alert in the largest group is lower than the minimum matching similarity θ_{alert}, the alerts are added as new keys in d (Lines 24-26) for finding matches in other groups.

After processing all groups, the algorithm iterates over all values of d, merges the alert lists using function *alert_merge*, and stores each of the generated merged alerts in the initially empty list h (Lines 28-32). List h thus contains a sequence of alerts merged from all groups $G \subseteq \mathcal{G}_\delta$, which means that list h is a meta-alert that has all properties of a group as required in Sect. 7.1.4. Finally, group h is returned by the function in Line 33.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Algorithm 5 Incremental merging.

```
1:  procedure ADD_GROUP(K_δ, g, θ_group,
            θ_alert, k_min, v_min, v_max)
2:      sim_max ← −1
3:      for all K_m ∈ K_δ do
4:          sim ←group_sim(g, m)
5:          if sim > sim_max then
6:              sim_max ← sim
7:              best ← m
8:              if sim = 1 then
9:                  break
10:             end if
11:         end if
12:     end for
13:     if sim_max < θ_group then
14:         best ←Meta_Alert()
15:     end if
16:     K_best ⇐ g
17:     K_best ←group_merge(K_best, θ_alert,
            k_min, v_min, v_max)
18: end procedure
```

Algorithm 6 Meta-alert generation.

```
1:  procedure GENERATE_META_ALERTS(A, Δ, θ_group,
            θ_alert, k_min, v_min, v_max)
2:      for all δ ∈ Δ do
3:          K_δ ← ∅
4:          for all A ⊆ A, ∀a ∈ A : ∃b ∈ A :
                    |a.t − b.t| < δ do
5:              g ← A : a_1.t ≤ a_2.t ≤ ··· ≤ a_n.t, ∀a_i ∈ A
6:              G_δ ⇐ g
7:          end for
8:      end for
9:      for all g ∈ G_δ do
10:         add_group(K_δ, g, θ_group, θ_alert, k_min,
                v_min, v_max)
11:     end for
12: end procedure
```

We do not provide the pseudo code for the generation of merged groups using the bag-of-alerts model for brevity. Similar to the group similarity algorithm from Sect. 7.3.2, the main difference is that alert representatives instead of the actual alerts are used for matching. In addition, intervals for occurrence counts of alerts are adjusted during merging so that all merged groups are appropriately represented (cf. Sect 7.3.4). The alignment of alerts in the bag-of-alerts model is computed by repeatedly applying the LCS procedure to the individual alignments of alert representatives of all groups. For more details on the realization of these methods, we refer to our implementation.

7.3.5 Meta-alert generation

The *group_merge* function presented in the previous section allows to generate meta-alerts from sets of similar groups. To select these groups, we outline procedure *add_group* in Alg. 5 that produces meta-alerts using the knowledge base as proposed in Sect. 7.2.1. In particular, the procedure iterates over all meta-alerts stored in the knowledge base (Line 3) and computes their similarities to the currently processed group g (Line 4) to find the meta-alert *best* that yields the highest similarity (Lines 5-11). In case that a comparison yields a perfect similarity of 1, there is no need to check all other meta-alerts and the loop stops prematurely to improve performance (Lines 8-10).

After the loop is completed, Line 13 checks whether the highest similarity between group g and any meta-alert is lower than threshold θ_{group} or no meta-alerts are available. In this case, a new meta-alert is generated by replacing *best* with an object of class $Meta_Alert$ in Line 14, otherwise *best* is an adequate match for group g. Either way, group g is added

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

to the knowledge base of meta-alert *best* (Line 16) and the corresponding meta-alert is subsequently updated (Line 10).

Function *generate_meta_alerts* in Alg. 6 runs the overall framework. The parameters involve all alerts $\mathcal{A}$, a group similarity threshold $\theta_{group} \in [0, 1]$, and the parameters $\theta_{alert}, k_{min}, v_{min}, v_{max}$ that are already known from Sect. 7.3.3 and Sect. 7.3.4. The function iterates over all $\delta \in \Delta$ values specified by the analyst (Lines 2-8) and initializes a knowledge base for every δ as an empty set (Line 3). Lines 4-7 represent the group formation phase for each δ value. Note that instead of the agglomerative clustering algorithm, we only display the requirements on the groups for simplicity, i.e., alerts in groups must occur in sufficiently close temporal proximity (Line 4) and be sorted by timestamp (Line 5). Finally, the function iterates over all groups and calls function *add_group* repeatedly (Lines 9-11). Note this procedure was designed for an offline setting where all groups are known in advance, however, we argue that it is easy to adapt the code for online analysis.

Note that the pseudo codes in this and the previous section leave out several aspects of our implementation that were omitted for brevity. This includes handling of multiple δ values by running function *generate_meta_alerts* in parallel, queuing strategies, and heuristics that allow to prematurely stop alert and group matching procedures for groups with low similarity to improve performance. Once more we refer to our implementation that provides more details on such aspects.

7.4 Evaluation

This section outlines our evaluation of the proposed alert aggregation approach. We first describe the methodology of our evaluation and introduce the data that we used to generate meta-alerts before showing and discussing the results. We point out that the code and data used in this evaluation are available open-source and thus all results are reproducible.

7.4.1 Methodology

The purpose of our evaluation is to validate the approach presented in this chapter with respect to well-known metrics that are relevant in machine learning and alert aggregation. Thereby, we use a publicly available real-world dataset described in Sect. 7.4.2 that centers around an illustrative attack scenario. The evaluation aims to demonstrate the capability of our framework to extract meta-alerts that represent attack manifestations present in the data.

For a better overview, we evaluate the introduced concepts and operations of our framework stepwise in alignment with Sect. 7.1. First, Sect. 7.4.3 provides empirical results from the group formation strategy and is used for the selection of appropriate δ values for the remainder of the evaluation. Section 7.4.4 contains a plot of the group similarities that allows to visually examine whether the proposed similarity functions are

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

suitable for allocating alert groups to attacks. The hierarchical clustering shown in Sect. 7.4.5 allows to draw similar conclusions, but additionally visualizes whether the outcomes of the proposed merging functions fit the overall picture. This hierarchical clustering also allows to estimate appropriate values for θ_{group}, which is used in the following sections.

The remaining evaluations focus on quantitatively measuring the performance of the overall approach rather than visual validation and parameter estimation. In Sect. 7.4.6, we execute the incremental alert aggregation procedure in a fully unsupervised setting and measure the clustering accuracy as well as the reduction rate. Section 7.4.7 on the other hand first generates meta-alerts in a supervised way and then measures the accuracy of classification of unknown sample alerts. We argue that both unsupervised and supervised evaluations are necessary to evaluate the capability of generating meta-alerts as well as using these meta-alerts for detection of similar alerts respectively.

We measure reduction rates for several δ values and thresholds $\theta_{group}, \theta_{alert}$ in Sect. 7.4.8. Since real systems are affected by false positive alerts that impair the performance of alert aggregation, we evaluate the robustness of our approach in Sect. 7.4.9. Moreover, we measure the runtime of our approach in Sect. 7.4.10. Finally, we discuss the results of the evaluation with respect to the requirements from Sect. 7.4.11. All evaluations are carried out on a 64-bit Windows 10 machine with an Intel i7-6600U CPU at 2.60 GHz and 16 GB RAM running Python 3.6.8.

7.4.2 Data

We use the publicly available dataset AIT-LDSv1.1 presented in Chap. 3 for our evaluation. The advantage of this dataset is that it comprises diverse log files collected from four different web servers that are targeted by the same attack scenario, which is a multi-step attack that involves (i) an Nmap [Nma] scan, (ii) a vulnerability scan using Nikto [Nik], (iii) an enumeration of user accounts using the `vrfy` command of the smtp-user-enum tool [Smt], (iv) a brute-force login attack using Hydra [Hyd], (v) an exploit of a webmail client for webshell upload (CVE-2019-9858), and (vi) an exploit of Exim for privilege escalation (CVE-2019-10149). The parameters of some attack steps are thereby varied so that their manifestations in the log data appear different on each system.

We used two open-source host-based intrusion detection systems, Wazuh [Waz] and AMiner [AMi], to process the logs and generate alerts. Wazuh is a signature-based detection engine that comes with a predefined set of rules and was used in its standard configuration. The sample alert from Fig. 7.2 is one of the alerts generated by Wazuh from the logs. AMiner on the other hand is an anomaly-based IDS that was configured to report unknown events as well as new values and combinations of values that occur in predefined positions of the log events. Since the exact execution times of each of the six aforementioned attack steps are provided in the dataset, we were able to label all alerts for our evaluation accordingly. Note that the generated alerts are not in IDMEF format or involve useful IP information and thus cannot be appropriately handled by existing approaches.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

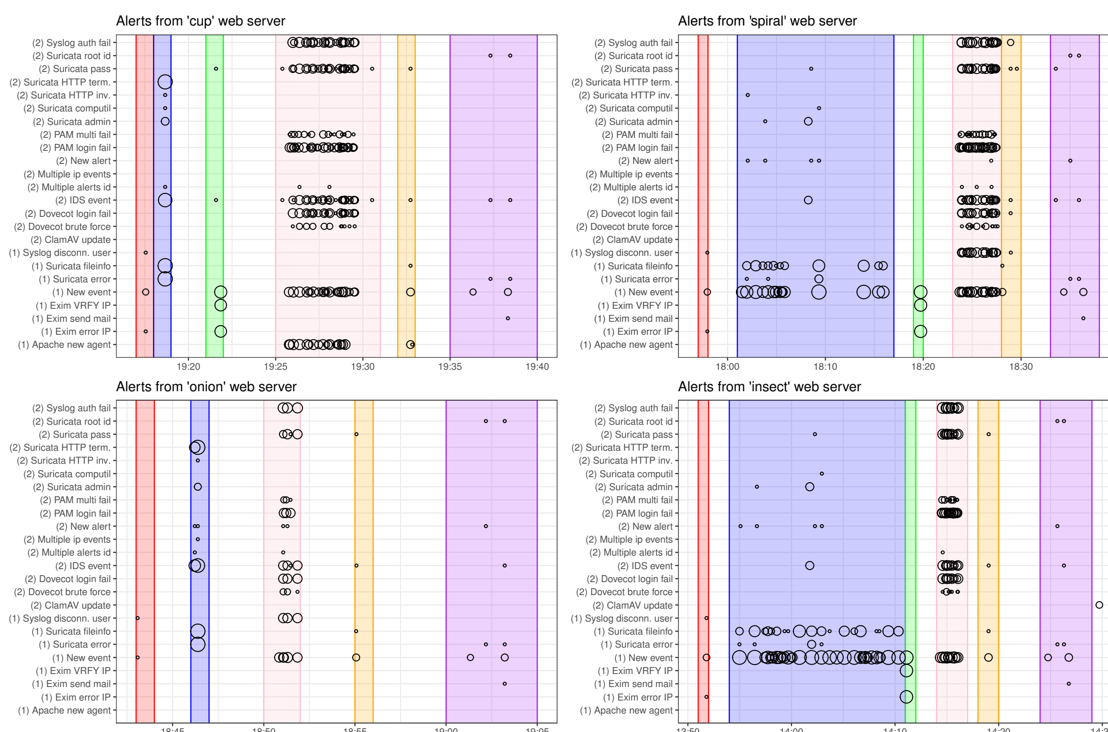


Figure 7.7: Alerts from AMiner (1) and Wazuh (2) IDS on four web servers. From left to right, the attacks are Nmap scans (red), Nikto vulnerability scans (blue), user enumerations (green), Hydra brute-force login attempts (pink), webmail exploits for webshell uploads (orange), and Exim exploits for privilege escalation (purple).

Figure 7.7 shows the alerts on timelines for each of the four web servers named *cup*, *spiral*, *onion*, and *insect*. The total number of alerts on all systems is 57,766. We display the type of the alerts (attribute "`AnalysisComponentName`" in AMiner alerts and attribute "`description`" in Wazuh alerts) on the vertical axis, where (1) marks AMiner anomaly types and (2) marks Wazuh rules. The alerts are displayed as circles, where larger sizes indicate more co-occurring alerts. These groups were formed by our grouping approach from Sect. 7.1.3 with $\delta = 1$ second. Note that the attack using the smtp-user-enum tool (green) was not executed on the *onion* web server and that there is a false positive alert caused by an update after the Exim exploit (purple) on the *insect* web server.

Comparing the graphs for the four web servers shows that attack parameter variations cause highly different alert patterns, in particular, the duration and amount of alerts generated by the Nikto scan (blue) and the user account enumeration (pink) varies greatly. On the other hand, close inspection of the plots reveal that some attack steps that are less affected by variations, e.g., the Exim exploit (purple), show the appearance of the same types of alerts with similar frequencies and timings on all systems. This suggests that it possible to derive meta-alerts across infrastructures that comprise attack patterns suitable for the detection of the same attack on other systems.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

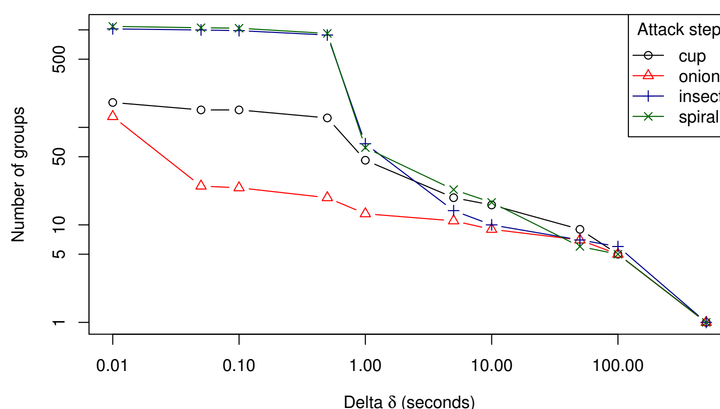


Figure 7.8: Influence of δ value on the number of generated alert groups on four systems.

7.4.3 Group formation

We discussed the importance of δ values for our grouping method in Sect. 7.1.3 and proposed to use multiple δ values in parallel to overcome issues with group formation in Sect. 7.2.1. To support this decision with real-world evidence, we plot the number of groups from four systems for different δ values in Fig. 7.8. Note that other than the plots in Fig. 7.7, groups are formed across all alert types.

As expected, fewer and larger groups are formed for increasingly larger values of δ and vice versa. In particular, the lowest selected δ value of 0.01 seconds yields a total of 2,417 groups on all systems, while a δ value of 500 seconds causes that only a single group on each system is generated, comprising all attack phases. This large range confirms that the usage of multiple δ values is reasonable. Furthermore, the figure suggests that δ values should be selected on a logarithmic range to avoid that the same or very similar groups are formed multiple times on different δ levels. Accordingly, we will only consider logarithmically distributed δ values in the following.

7.4.4 Group Similarities

Due to the fact that our meta-alert generation approach is based on group similarities, it is necessary that our similarity functions are capable of clustering related groups with high accuracy. We therefore compute a pairwise similarity matrix of all groups formed using a specific δ value. For this, we make use of the *group_sim* function (cf. Sect. 7.3.2) that relies on the alert similarity function *alert_sim* (cf. Sect. 7.3.1). As outlined in Sect. 7.1.4, quadratic runtime complexity of the exact matching strategy makes it necessary to switch to the bag-of-alerts strategy for large groups. We empirically determined that $l_{bag} = 2000$ keeps processing times for most groups below 0.05 seconds, which we consider acceptable. Furthermore, we set $k_{min} = 0.1$, $v_{min} = 0.1$, and $v_{max} = 10$ to ensure that meta-alerts do not contain alert characteristics that occur in less than 10% of merged groups. Finally, we set the weight of alignment information to 0.1 and the weight of

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

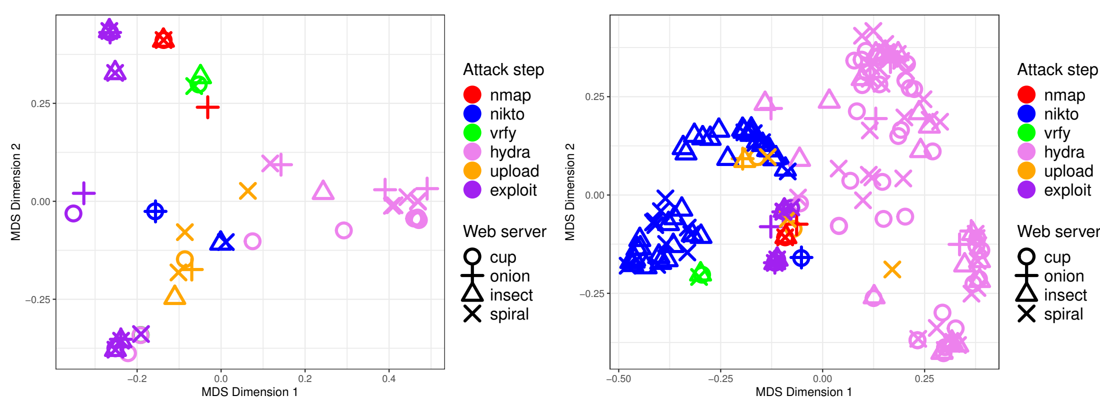


Figure 7.9: Multi-dimensional scaling of pairwise group similarities using δ values of 10 seconds (left) and 1 second (right) shows that groups of same attack steps are frequently located close together.

attribute "timestamp" to 0. These parameters are used in all following evaluations unless stated otherwise.

We then apply multi-dimensional scaling [BG05], which is a dimensionality-reduction technique that allows to represent groups as points that largely retain their original similarities derived from the pairwise similarity matrix. Figure 7.9 shows the groups formed with δ values of 10 seconds (left) and 1 second (right) plotted on the first two dimensions. Groups are marked with the same symbol if they originate from the same system and have the same color if they occurred in the same attack phase. Since groups that relate to the same attack should be similar independent from the system infrastructure, we expect to see groups with the same color and different symbols close together.

The left figure shows that several groups correctly form clusters of related attack steps. For example, the groups related to the "vrfy" attack are placed close together and are relatively isolated. Other groups, such as the ones belonging to the "hydra" attack, appear more spread out. The groups belonging to "nikto" result in two distinct clusters, which corresponds to Fig. 7.7 that shows that this attack step lasted over a long time on *spiral* and *insect* systems, but only a short time on *cup* and *onion*. Similarly, the "exploit" attack step forms four separate clusters. The reason for this is that this attack step actually consists of several smaller steps that are sequentially executed and disclosed as separate groups at this δ level, but comprise rather different alert types and frequencies.

The right plot of Fig. 7.9 shows that much more groups are generated for the "nikto" and "hydra" attack when δ is set to a lower value. Since groups that belong to one of these two attacks dominate the variance of the data, it is difficult to reason about the correct clustering of groups that belong to other attacks. However, the fact that groups of "nikto" and "hydra" attacks are clearly separated suggests that similarity-based group clustering is reasonable even for small δ values.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

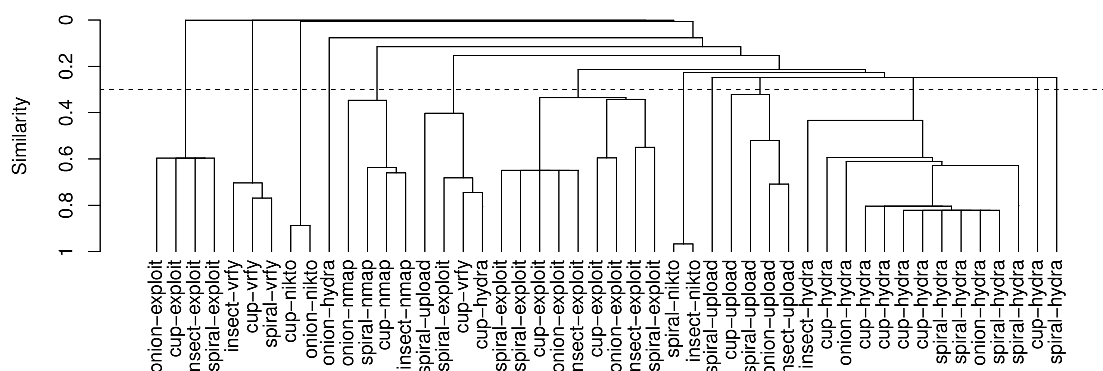


Figure 7.10: Dendrogram of group similarities, where each node represents a merged group. The dashed line shows a reasonable cutoff threshold at a similarity of 0.3 that yields meta-alerts with few incorrect allocations.

7.4.5 Hierarchical Aggregation

We evaluate our merging functions *alert_merge* (cf. Sect. 7.3.3) and *group_merge* (cf. Sect. 7.3.4) by first computing the pairwise similarity matrix and then merging the two groups that yield the highest similarity. These groups are then removed from the similarity matrix and instead the resulting merged group is added by computing its similarity to all remaining groups. This is repeated until the similarity matrix only contains one group. At this point, we are able to construct a rooted dendrogram with branches that connect at the height of the similarity of the merged groups.

Figure 7.10 displays such a dendrogram for groups formed with a δ value of 10 seconds. The original groups are placed at the leaves of the tree and are labeled with the names of their corresponding systems and attack steps, connected with a dash. Each node of the tree represents a merged group, where the height displayed on the vertical axis describes the similarity between the groups used for merging. Examining the dendrogram shows that groups that are part of the same attack phase are frequently merged with a relatively high similarity, while groups that relate to different attacks only merge with low similarity. For example, each of the first four "exploit" groups from the left occurred on different systems and were merged with a similarity of 0.6. The similarity of the resulting merged group to all other groups was 0, indicating that meta-alerts do not tend to over-generalize.

To select the similarity threshold θ_{group}, we plot cluster purity [MSR08] against the number of clusters in Fig. 7.11. The similarity threshold is selected so that purity is large, i.e., clusters contain mostly groups belonging to the same attack phase, and the number of clusters approximates the true number of attack steps, which is 6 in our case. The figure shows that purity drops for thresholds lower than 0.25, while the number of clusters continuously increases for larger thresholds. Accordingly, we select $\theta_{group} = 0.3$ as a reasonable trade-off that yields a purity of 0.94 and 13 clusters.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

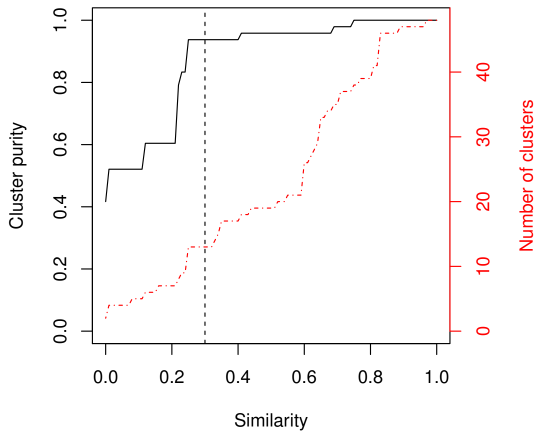


Figure 7.11: Cluster purity and the total number of clusters suggests 0.3 as a similarity cutoff threshold.

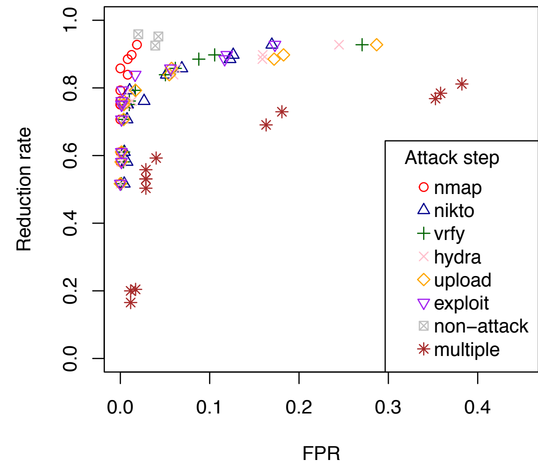


Figure 7.12: Reduction rate plotted against false positive rate for several threshold values.

The reason why the number of clusters is larger than the number of attack phases is due to the fact that the labels of the data are not sufficiently fine-grained, i.e., some attack phases actually comprise sequences of sub-steps that should be labeled differently from each other. We decided against manually altering the ground truth data to fit our needs and we will therefore mainly focus on the correct separation of groups into homogeneous meta-alerts in the next section.

7.4.6 Meta-Alert Generation

The previous sections evaluated the similarity and merging functions. In this section, we evaluate the procedure using these functions for incremental meta-alert generation (cf. Sect. 7.3.5). For this, we use the logarithmic storage strategy with a maximum queue size of 25. We then create groups for intervals $\Delta = \{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500\}$ seconds and iteratively generate meta-alerts through continuous similarity computations and merging. To analyze the influence of the thresholds specifying minimum similarity for group merging and alert matching, we conduct the experiment multiple times and vary $\theta_{group}, \theta_{alert}$ equally in the range of $[0.1, 0.7]$ and a step width of 0.05.

We obtain a set of meta-alerts $\mathcal{M}_\delta$ for each $\delta \in \Delta$ once all groups are processed. To evaluate the quality of the meta-alert generation procedure, we measure the homogeneity of all meta-alerts. In particular, we count (i) two groups with the same attack phase label in the same meta-alert as true positives (TP), (ii) two groups with different labels in the same meta-alert as false positives (FP), (iii) two groups of the same label in different meta-alerts as false negatives (FN), and (iv) two groups with different labels in different meta-alerts as true negatives (TN). As mentioned in previous sections, groups which belong to the same attack phases frequently end up in separate meta-alerts due to sub-

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

steps in attack executions, and the true positive rate $TPR = TP/(TP + FN)$ is therefore not expressive. For this reason, we plot the false positive rate $FPR = FP/(FP + TN)$ against the reduction rate, i.e., the ratio between the number of meta-alerts and groups computed as $r_{group} = |\mathcal{M}_\delta| / |\mathcal{G}_\delta|$. To deal with groups that only contain false positive alerts and groups that span over several attack phases, we introduce the labels *non-attack* and *multiple*.

Figure 7.12 shows the results of aforementioned calculations carried out for each attack phase separately. Each point shows the average FPR and reduction rate over all δ values achieved for a particular threshold for both θ_{group} and θ_{alert}. In general, larger thresholds yield more meta-alerts that lead to lower FPR and reduction rates, i.e., points closer to the bottom-left of the plot, while smaller thresholds cause that groups are more easily merged to meta-alerts, which increases reduction rate and FPR, i.e., result in points closer to the top-right of the plot. There is thus a trade-off between the reduction rate and accuracy. Thresholds in the range $[0.2, 0.4]$ yield the best results for all attack phases and achieve average reduction rates of around 80% and average FPR of less than 5% for all attack types. These results correspond to a rule of thumb from Husák et al. [HČLV17], who state that up to 85% of alerts can reasonably be aggregated on average.

7.4.7 Cross-System Classification

The previous section focused on the evaluation of the incremental meta-alert generation procedure in an unsupervised way. To evaluate whether the generated meta-alerts are suitable for classification of attack executions on other systems, we carry out a supervised evaluation. In particular, we use alerts from three out of the four systems to form groups and generate meta-alerts in a training phase, where groups are only allocated and merged with meta-alerts that belong to the same attack phase. We then use the alerts from the fourth system as test data to generate groups, determine the most similar meta-alert, and measure the accuracy of this classification. We also require that the similarity to the best-matching meta-alert exceeds 0.1, otherwise the group is assigned to the non-attack class. Thereby, we count (i) a true positive (TP) for an attack if a group belonging to that attack is correctly allocated to a meta-alert of the same attack, (ii) a false positive (FP) for the meta-alert's attack and (iii) a false negative (FN) for the group's attack if the group is incorrectly allocated to a meta-alert with a different attack, and (iv) a true negative (TN) for all attacks that the group is correctly not assigned to. To obtain better estimations for model performance through cross-validation, we repeat this procedure so that alerts from every system are used as test data and average the results. As before, we also use a range of δ values and compute all rates as averages.

The left side of Fig. 7.13 shows TPR plotted against FPR for all attacks, where each point represents the results achieved using a specific threshold. The graph shows that each of the six original attack classes achieve a low FPR of less than 5%. The TPR appears to depend on the attack type, since several points that refer to the same attack label are relatively close together and form groups. The attacks achieve TPR in the range $[0.75, 0.95]$ for most threshold settings, except for the "upload" attack, which performs

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

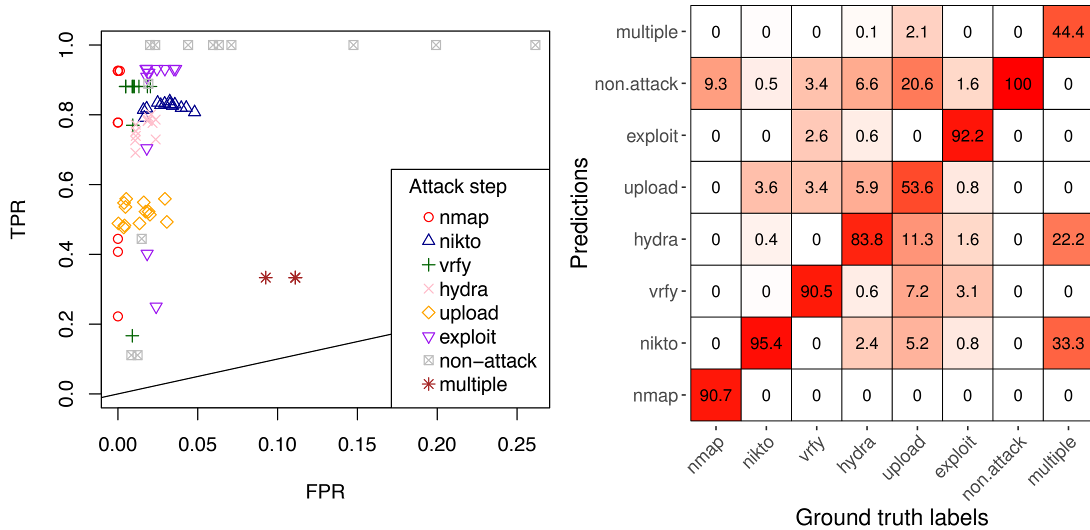


Figure 7.13: Left: True positive rate plotted against false positive rate of cross-system attack classifications for several threshold values. Right: Confusion matrix of attack classifications using 0.3 as threshold.

comparatively bad with a TPR of only 0.5. The reason for this is that this attack only caused similar alert sequences in *onion* and *insect*, but involved additional alerts in *cup* and *spiral* due to differences in the infrastructure setup (cf. Fig. 7.7).

The right side of Fig. 7.13 shows the confusion matrix for $\theta_{group} = \theta_{alert} = 0.3$. Note that other than the plot on the left side, the confusion matrix represents total numbers of TP, FP, FN, and TN rather than averages over all δ values. We normalized the matrix column-wise to obtain the relative frequencies of class allocations so that TPR is visible in the main diagonal. This allows to obtain a better overview of the misclassifications, e.g., around 50% of alert groups belonging to "upload" are incorrectly classified as one of "non-attack", "vrfy", "hydra", or "nikto" for aforementioned reasons.

We also visualize the performance of our classifier with respect to δ. For this, we use the F1-score computed by $F1 = TP/(TP + 0.5 * (FN + FP))$, since it provides a single measure that is large when both FN and FP are low. Figure 7.14 shows box plots of the F1-score for several δ values, split up by attack steps and thresholds. As visible by the height of the boxes, the variance of the F1-score with respect to δ values is relatively small. In addition, four out of six attacks have at least one setting for the threshold and δ value so that the highest possible F1-score is reached. The plot also confirms that the performance is mostly dependent on the type of attack, since similar F1-scores are reached for most threshold values. There are some exceptions to this observations, in particular, the performance declines for a threshold of 0.7 for "vrfy" and for thresholds larger than 0.6 for "exploit" and "nmap". As visible in Fig. 7.13, this is due to a decrease of TPR.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

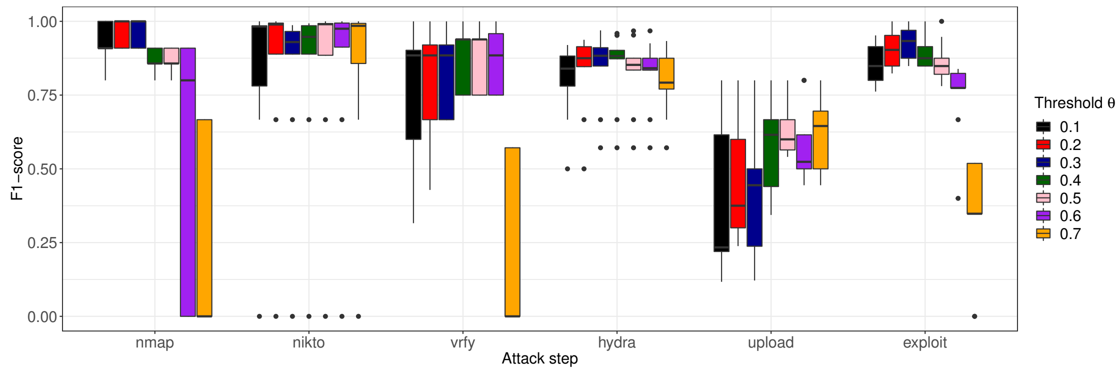


Figure 7.14: F1-score boxplots of cross-system classification for several δ values, categorized by attack and θ_{group}.

7.4.8 Reduction

The ability to reduce alert groups to meta-alerts is a key feature of our approach, because a high reduction rate indicates that many groups were merged to few meta-alerts. As shown in Fig. 7.12, there is a limit to the reduction rate at which accuracy starts to decline. To improve understanding of parameter influence on the reduction rate, we therefore visualize the reduction rate for several combinations of δ values and thresholds $\theta_{group}, \theta_{alert}$ in the following. Note that we compute the reduction rate on alerts of only four systems. Using more data, i.e., obtaining alerts of the same attack scenario from additional systems, would likely increase the reduction rates.

The left plot in Fig. 7.15 shows the group reduction rates, i.e., the ratio between the number of meta-alerts and the number of groups computed as $r_{group} = 1 - |\mathcal{M}_\delta| \,/\, |\mathcal{G}_\delta|$. The plot shows that reduction rates decrease for increasing thresholds, because larger thresholds mean that it is less likely that groups reach the minimum required similarity to be allocated to meta-alerts. Moreover, reduction rates decrease for increasing δ values, because smaller δ values cause that more groups are generated, making it easier to find similar groups. For example, the "hydra" brute-force attack repeats the same action multiple times in short intervals, and smaller δ values break up the generated alert patterns into shorter sequences that are then suitable to be merged with each other. A cutoff appears around $\delta = 0.5$ seconds that reaches reduction rates of around 88% to 99% for all thresholds, while $\delta = 1$ second yields reduction rates down to 33% for high thresholds.

The right plot in Fig. 7.15 shows the reduction rates of alerts rather than groups. The value is computed as the average reduction rate of all meta-alerts, i.e., $r_{alert} = (1/\,|\mathcal{M}_\delta|)\,\Sigma_{m \in \mathcal{M}_\delta}\,(1 - |m|\,/\,(|K_m|))$. Overall, the alert reduction rates also decrease for increasing δ values and thresholds, however, to a less effect compared to the group reduction rates.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

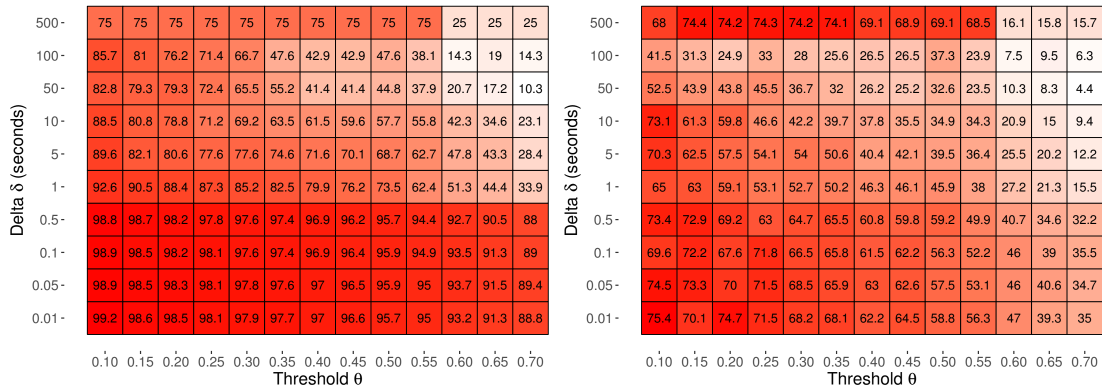


Figure 7.15: Influence of δ and θ_{group} on reduction rates. Left: Group reduction rates. Right: Alert reduction rates.

7.4.9 Robustness

Our approach relies on the assumption that adequate alert groups are formed in the first stage of our procedure (cf. Sect. 7.1.3). Despite using several δ values in parallel, the grouping phase is susceptible to intervening alerts that are not part of attacks. In particular, such noise alerts form new groups, change the composition of existing groups, or cause that groups are combined.

To evaluate the robustness of our approach with respect to noise alerts, we randomly duplicate alerts and uniformly distribute them over the input data. Adjusting the total number of alerts added in this way allows us to set the noise intensity. The plots in Fig. 7.16 show the influence of noise alerts on the number of generated groups and classification performance on each system, where the noise intensity on the horizontal axis is displayed as the average amount of alerts inserted per minute. As visible in the plot on the left side, the total number of groups increases, since random alerts that occur with a temporal distance larger than δ to other alerts form new groups. The curve peaks when approximately 10 noise alerts are inserted, followed by a rapid decline caused by group merges. This is reasonable, since 10 noise alerts per minute mean that an alert is inserted every 6 seconds on average, which corresponds to the used δ values of 5 seconds. The plot on the right side shows that the average TPR and F1-score decline with increasing noise intensity, while the FPR remains constant at a low level. In particular, TPR and F1-score rapidly decrease from around 0.6 to 0 approximately when 10 noise alerts per minute are inserted, corresponding to the peak of the number of groups. We therefore conclude that δ functions as a breakdown point for our approach and that δ values exceeding the average noise intensity should be avoided.

To overcome this issue, we recommend to reconfigure the deployed IDSs. Randomly and repeatedly occurring alerts indicate that some sensors are too sensitive and therefore report normal behavior as malicious, which is not desirable for manual or automatic analysis and should be fixed anyway. Alternatively, it is also possible to set up a

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

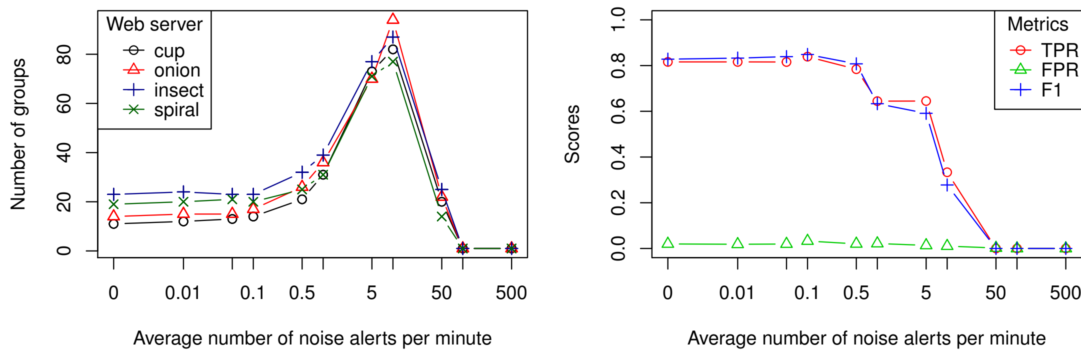


Figure 7.16: Robustness to randomly generated noise alerts for $\delta = 5$ seconds and $\theta_{group} = \theta_{alert} = 0.3$. Left: Influence on number of alert groups. Right: Influence on TPR, FPR, and F1-score.

filter for particular alert types that are known to report false positive alerts prior to performing alert aggregation. In particular, meta-alerts generated by our approach that have been manually labeled as false positive alerts could be used to design such filter rules. Furthermore, our implementation also provides a non-symmetric variation of the group similarity metric outlined in Sect. 7.3.2 that measures how well one alert group is represented by another and thus improves robustness against noise alerts in one of the groups.

7.4.10 Runtime

We analyze the runtime of our approach by measuring the time it takes to process groups, i.e., compute the similarity between a group and all meta-alerts, find the best matching meta-alert, add the group to the knowledge base, and generate the meta-alert. Since our procedure is incremental, the number of meta-alerts and therefore also the number of necessary similarity computations is increasing over time, causing that the processing time per group is also expected to increase. However, due to the fact that most meta-alerts are generated at the beginning and few new meta-alerts are generated over the long run, the runtime should be approximately linear.

In the following, we considered alerts rather than groups to compensate for the fact that larger groups likely require more time to process than smaller groups. Figure 7.17 therefore shows the cumulative runtimes it took to process alerts for several δ values at thresholds $\theta_{group} = \theta_{alert} = 0.3$ (left) and different thresholds $\theta_{group}, \theta_{alert}$ using $\delta = 0.1$ seconds (right). Note that for $\delta > 0.01$ two large groups are generated by “nikto” on systems cup and onion, each with approximately $20,000$ alerts. The plots show that processing these large groups (approx. alerts 0 to $40,000$) as well as large amounts of small groups (approx. alerts $40,000$ to $60,000$) largely follows linear complexity.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

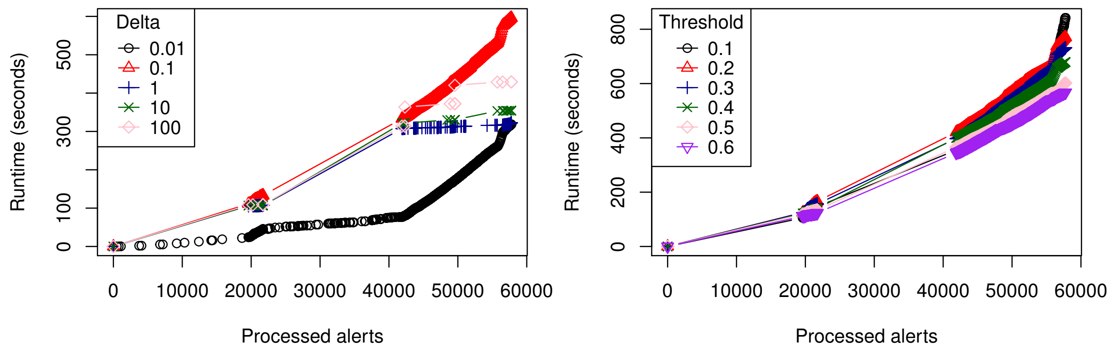


Figure 7.17: Runtime required to process alerts for different values for different δ (left) and thresholds (right).

7.4.11 Discussion

The previous sections present empirical results that give insights into the practical application of the proposed alert similarity functions and aggregation strategies. The results assert the functioning of the approach and confirm the fulfillment of all requirements on a domain-independent alert aggregation approach stated in Sect. 2.1. Requirement (1) is fulfilled, because our framework is fully automatic and capable of extracting meta-alerts representing unknown attack scenarios. We recognize that sequential-based and case-based methods that rely on manually coded knowledge are capable of modeling only the most distinct features of attacks and thus achieve higher accuracy when classifying attack executions than similarity-based methods, especially when variations of attacks or IDS configurations are considered. However, we argue that this issue generally applies to all similarity-based methods and is compensated by the ability to detect new attacks. Furthermore, meta-alerts generated by our approach could ease the process of manual attack specification, since the merging procedure also reduces attributes and values of alerts as well as alert occurrences to typical properties of the attacks.

Our approach combines time-based and attribute-based grouping strategies by contextualizing alerts through temporal proximity and considering all available attributes for similarity computation. This solves the problem of mapping alerts to attacks in alignment with requirement (2). Due to our format-agnostic similarity metrics, requirement (3) is also fulfilled.

To meet requirement (4), we designed our approach as an incremental clustering procedure and avoided over-generalization of meta-alerts by the use of knowledge bases. Queueing strategies (cf. Sect. 7.2.1) thereby ensure that the time required to update meta-alerts is not continuously increasing. We recorded the processing times during our evaluations and ascertained that the overall runtime is approximately linear with respect to the number of alerts processed.

Finally, we see our generated meta-alerts as improvements over state-of-the-art that usually involve graphs of attack steps, because they have the same semi-structured format

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

as incoming alerts and are therefore easy to understand for humans and support machine processing. Regarding requirement (5), our approach is thus implementing a combined strategy of meta-alert representation, since single alerts are enriched with mergelists and wildcards, and embedded in sequences.

7.5 Summary

In this chapter we introduced a novel approach for meta-alert generation based on automatic alert aggregation. Our method is designed for arbitrary formatted alerts and does not require manually crafted attack scenarios. This enables to process alerts from anomaly-based and host-based IDSs that involve heterogeneous alert formats and lack IP information, which is hardly possible using state-of-the-art methods. We presented a similarity metric for semi-structured alerts and three different strategies for similarity computation of alert groups: exact matching, bag-of-alerts matching, and alignment-based matching. Moreover, we proposed techniques for merging multiple alerts into a single representative alert and multiple alert groups into a meta-alert. We outlined an incremental procedure for continuous generation of meta-alerts using aforementioned metrics and techniques that also enables the classification of incoming alerts in online settings.

For our evaluation, we generated alert datasets by forensically analyzing the log datasets presented in Chap. 3 using a signature-based IDS and an anomaly-based IDS. The evaluation shows that our approach is capable of reducing the number of alert groups by around 80% while maintaining a true positive rate of around 80% and a false positive rate of less than 5%. These results suggest that our framework is suitable for application in real-world scenarios, in particular, situations where a large number of similarly configured machines are available and meta-alerts could be generated and used for detection across systems with high precision.

We consider the meta-alerts generated as a result of our approach as cyber threat intelligence (CTI) as defined in Sect. 2.3. In particular, the meta-alerts are capable of expressing complex behavior patterns manifesting in log data as sequences of alerts generated by IDSs. Thereby, a key feature is that meta-alerts are detectable by comparing newly observed alert sequences with the patterns stored in each available meta-alert. As such, the meta-alerts combine the advantages of abstract TTPs and measurable IoCs. Since meta-alert generation is accomplished without any manual or forensic analysis, the meta-alerts are also immediately actionable, i.e., may be used to disclose similar attacks or at least make analysts aware of suspicious system behavior. Nonetheless, to actually classify unknown alert sequences, it is obviously necessary to have a human-in-the-loop who assigns labels to respective meta-alerts that should be used to recognize similar attacks on other systems.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 8

Conclusion

The thesis proposed a procedure for extracting cyber threat intelligence (CTI) from raw log data. Thereby, one of the most crucial aspects turned out to be the availability of labeled training data that is representative for diverse system and network infrastructures. For this purpose, log data is generated and labeled in testbeds to enable repeated attack execution with variations in controlled environments. The generated data is then parsed and analyzed for anomalies using a custom detection method for categorical values, which are common in log data. Subsequently, anomalies and alerts reported by such detectors are aggregated into detectable meta-alerts that allow recognition of the same attacks on other systems. The contributions presented in this dissertation address the problem statements and research questions stated in Sect. 1.1. In the following, the answers to the research questions are summarized.

RQ1: What is an appropriate method to enable flexible generation of realistic and labeled log data that facilitates evaluation of CTI extraction approaches?

It may seem that the ideal case is to collect log data from productive system infrastructures of many different organizations and employ human specialists to manually label all normal and malicious activities in the logs. However, this is hardly feasible in practice, as log data from real infrastructures is usually very sensitive and manual label assignment is difficult to carry out reliably in large amounts of logs. Fortunately, testbeds are a viable alternative to real system and network infrastructures. The strong benefits of testbeds include the possibilities to run simulations of normal behavior and attacks in isolated settings so that no unknown activities are incorrectly labeled, to control all parameters of the testbed and simulation, and to generate arbitrary numbers of testbeds. In particular, the latter aspect is enabled by model-driven concepts that introduce variations in every instantiated testbed as described in this thesis. This is essential to generate log data that is representative for diverse infrastructures and attack executions, which is required to evaluate whether extracted CTI enables detection of same or similar attacks in different system environments. Chapter 4 demonstrates that the semi-synthetic log data is similar

to logs generated by real users in terms of access frequencies and behavior patterns. In addition, the evaluation presented in Chap. 3 indicates that the shift from traditional testbeds to abstract models of testbed components eases adjustments or extensions of testbeds and thus enables flexible testbed development.

RQ2: Which methods are appropriate to parse raw and unstructured log data to enable unsupervised event classification, parameter analysis, and incremental anomaly detection?

Most of the time, log data occurs as sequences of heterogeneous events, i.e., there are multiple event types with distinct syntaxes that need to be considered to process the whole log file. While the static parts of a log line determine the event type, the variable parts (i.e., the parameters) need to be extracted for analysis. As many log events share some of the static parts, the most efficient way to process logs are parser trees, i.e., data structures where nodes that connect in sequences or branches represent either static or variable parts of the lines. The survey carried out in Chap. 5 shows that there are many different techniques for generating such parser trees automatically; most common are similarity-based techniques that split lines into tokens using delimiters (e.g., white space) and then apply string metrics. An alternative to that are density-based techniques that consider frequent tokens as static and others as variable. The survey also showed that there are several classes of algorithms for parser generation that are designed for specific requirements, e.g., forensic or online analysis, character- or token-based event processing, static or adaptive clustering, single or multiple event analysis, etc. When parsers are generated from training log files, parsing itself becomes a simple form of anomaly detection, since structures and parameters of new logs that do not fit the learned event types are detected as outliers. Once data is parsed, subsequent detection techniques usually either focus on extracted values or analyze the sequences of event types.

RQ3: To what extent can a detection method be designed to analyze categorical values in parsed log data for the purpose of identifying anomalous system states with high accuracy?

Reviewing the state-of-the-art shows that there are several different methods to detect anomalies in log data. This is reasonable, since log data exhibits plenty of characteristics that may be affected by attacks, including event correlations, sequences, and timing, as well as all patterns that occur in their parameters, e.g., statistical properties in numeric values. The anomaly detection technique presented in Chap. 6 focuses on parameters for categorical values, which are common in logs but often difficult to analyze. In particular, some categorical parameters correlate with each other (i.e., the likelihood that one value occurs depends on another value in the same event), however, there are too many possible combinations to consider all pairs of parameters. The proposed solution is a sequence of filtering steps that select correlating values for further analysis based on several factors, including their number of unique values, similarities of independent or conditional value probability distributions, and correlation strengths. Statistical tests are used to detect anomalies as changes of usual correlation behavior that persisted over a long time. The evaluation of the presented detection technique shows that it is capable of recognizing even small changes of value distributions as well as changes that affect only rarely occurring values with a true positive rate of 80% and a false positive rate of less than 5%.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

RQ4: What is an appropriate method to technically describe advanced attacks involving multiple consecutive attack steps as artifacts and complex system behavior patterns in an abstract way to provide reusability on other systems?

One of the main problems is that there exists no unique mapping between alerts and attacks, as different alert sequences are generated by single attack executions but some of these alerts may occur also as part of other attacks. One way to tackle this problem is to focus on sequences of alerts as they are more distinct for certain attacks than individual alert instances. Another problem is that the attributes of alerts are unknown beforehand as different IDSs do not use the same alert formats. The solution proposed in this thesis therefore leverages a similarity metric that is independent of any domain knowledge about alerts or monitored systems, which makes it more widely applicable. The approach assumes, however, that alerts that are related to each other occur in close proximity, which is necessary to form groups. Extending the similarity metric to these groups enables to continuously merge alert groups from diverse systems to generate abstract meta-alerts. Since it is possible to compute the similarity between any of the meta-alerts with newly observed alert sequences on another system, they are reusable for attack classification. The evaluation presented in Chap. 7 showed that the approach yields true positive rates of around 75%-95% depending on the attack type and reduce the number of alerts by up to 85%.

There are many ways how the concepts proposed in this thesis may be extended in future work. Foremost, the model-driven approach for testbed generation allows to target many additional use-cases other than enterprise networks, for example, use-cases centered on Internet-of-Things (IoT). Independent of the use-case, a problem that remains with testbeds is that simulations need to run in real-time, which makes it difficult to generate log data that covers large periods of time, e.g., months or years. This issue could possibly be alleviated by training Generative Adversarial Networks (GAN) on log data from testbeds and use them to generate more logs that follow the same patterns and exhibit similar characteristics; a technique that has already been demonstrated for network traffic [SBJ+20] and could be extended to system log data. Thereby, testbeds with variations could be particularly valuable to provide a diverse baseline for training. Another useful addition to testbeds would be an approach for (semi-)automatic generation of labeling rules from observed malicious log events. Such a mechanism could largely reduce the amount of time spent by analysts for modeling testbeds, specifically when attacks are adjusted causing their consequences in log data to change, and thus labeling rules need to be updated. One way to achieve this could be to automatically execute attacks with all (or sufficiently many) combinations of parameter values in an idle testbed where no other actions are carried out, capture their consequences in log data, and derive a small set of identifiers that discerns them from the logs corresponding to normal behavior.

When it comes to anomaly detection methods for log data, the possibilities are virtually endless, as log data has so many different inherent regularities that ask for the development and application of new techniques for pattern extraction. The main recommendation

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

is to design detection techniques that are appropriate to process log data as streams, i.e., they need to be capable of efficient online or incremental learning and utilize a dynamic baseline for detection to comply with changing system infrastructures. One of the most challenging aspects of log analysis is thereby the breadth of events formats and their unique characteristics. As requirements for logging are just too versatile, it is unreasonable to expect that a unified log format will ever gain universal acceptance. Accordingly, advanced analysis methods are often either too generic to leverage specific properties of certain types of log data, or too specific to be broadly applicable. There is therefore a need to find solutions in between those two extremes, in particular, anomaly detection methods for specific types of log files that yield very low false positive rates independent of system usage to ensure applicability in productive systems. Another interesting research problem is anomaly detection using federated learning [PRT+18], i.e., extending the training across many systems to obtain more generic models and reduce the required training time. Since the testbeds generated as part of this dissertation are representative for diverse systems, they could be beneficial to evaluate intrusion detection relying on federated learning.

The aforementioned problems regarding diverse log formats directly propagate to alert aggregation as log events are at the core of alerts and anomalies. This thesis presented an alert aggregation approach that is independent from alert formats, but still relies on certain properties of the data, in particular, alerts in JSON format. Extensions to this approach could either generalize this idea for other formats or even natural language, which is common in reports of cyber incidents that often contain information on artifacts that are useful for detection. Moreover, our approach could be extended to implement a function that measures how well one alert group is contained in another one. This could reduce the problem of noise within groups and could even be used to separate alerts of overlapping attack executions into distinct groups. On the other hand, determining how well meta-alerts are represented by groups could allow to automatically recognize and improve incorrectly formed meta-alerts. Finally, the presented approach generates a hierarchical structure of groups due to the fact that different interval times are used for group formation. It could be interesting to transfer these relationships between groups to meta-alerts in order to improve their precision.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Figures

1.1 Concept for sharing anomalies to enable automatic attack detection and classification across organizations. 2
1.2 Pipeline of the proposed approach for CTI extraction from raw log data. . 9

3.1 Model-driven testbed generation approach. Testbed-independent models (TIM) are derived from real scenarios and transformed to testbed-specific models (TSM) that instantiate the testbed for labeled network and log data generation. 37
3.2 Simplified sample transformation of TIM (left) to TSM (right) for infrastructure setup. 38
3.3 Simulated normal user behavior on the Horde Webmail (yellow) and OkayCMS (blue) modeled as a state machine. 40
3.4 Randomized user profiles and state machine of system behavior TIM (left) transformed to TSM (top right) and testbed execution (bottom right). . . 40
3.5 Multi-step attack on Horde Webmail (top) and its transformation from TIM (center) to TSM (bottom). 41
3.6 Example of our labeling procedure. Information on attack execution (top left) and expected attack logs (top right) are used to create labels (bottom left) using time-based and line-based techniques for log data (bottom right). . 43
3.7 Technical implementation of the model-driven testbed and log data generation approach. Simple arrows indicate imports and filled arrows indicate generation of resources such as scripts, configuration files, or machines. 44
3.8 Overview of role dependencies. To install any role, it is necessary that all roles that the attached arrows point to are available and correctly installed. 45
3.9 Event frequencies of Apache access logs. Scans executed as part of a multi-step attack manifest themselves as peaks (shaded intervals). 47
3.10 Biplot of user page visit frequencies aggregated in daily intervals. 48
3.11 Biplot of Apache access logs collected from different testbeds. 48

4.1 Labeling concept for model-driven testbeds. 55
4.2 Overview of the testbed network. Steps (1)-(3) mark the attacker's path to compromise the intranet server and steps (a)-(c) represent connections related to the data exfiltration attack vector. 57
4.3 User state machine for simulating normal behavior on the cloud share platform. 58

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.4 Query labeling rule that matches domain names. 65
4.5 Labeling rule of sequence type with filtering. 66
4.6 Labeling rule of sub query type. 67
4.7 Labeling rule of parent query type. 68
4.8 Procedure for labeling rule type selection. 69
4.9 Event counts in DNS logs for different services. 74
4.10 Apache access logs with attack consequences of scans. 74
4.11 Monitoring logs of CPU (top) and memory (bottom) showing attack consequences of password cracking. 75
4.12 Different log events caused by the attacker escalating to system privileges in the *fox* (top) and *harrison* (bottom) datasets. 76
4.13 Occurrences of events labeled as part of the attack steps. 77
4.14 Euler diagram depicting label relationships. 80
4.15 Survey question asking the participant to decide if the marked line is correctly labeled as part of the dirb attack step. 81
4.16 Boxplots of survey answers to correctly labeled lines show that participants mostly agreed with assigned labels. 82
4.17 Boxplots of survey answers to incorrectly labeled lines show that participants recognized consciously placed errors. 82

5.1 Sample log messages for static analysis. 92
5.2 Sample log messages and their event allocations for dynamic analysis. . . 93
5.3 Sample log events visualized on a timeline. 93
5.4 Relative frequencies of static clustering techniques used in the reviewed articles. 104
5.5 Relative frequencies of benchmarks used for evaluation. 110
5.6 Relative frequencies of log data used for evaluation. 113

6.1 Procedure of the Variable Type Detector. Correlations between variables and values are filtered iteratively. 120
6.2 Comparison of VCD selection with association metrics. 129
6.3 Anomaly detection ROC plots for two attack scenarios. 130
6.4 Value co-occurrences of damped correlation. 132
6.5 Influence of thresholds on accuracy of correlation selection. 133

7.1 Overview of the relationships between concepts. Alerts (top) occurring on timelines (t) are grouped by temporal proximity (center) and then aggregated to meta-alerts by similarity (bottom). 140
7.2 Simplified sample alert documenting a failed user login. 142
7.3 Alert occurrences duplicated over four parallel timelines show the formation of alert groups based on alert interval times. Larger intervals (top) allow more elapsed time between alerts and thus lead to fewer and larger groups compared to smaller intervals (bottom). 144

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

7.4 Merging strategies for alert groups. Left: Finding exact matches between alert pairs. Center: Matching representatives using a bag-of-alerts model. Right: Matching using alert sequence alignment. 148
7.5 Scenario for cross-system alert recognition. Steps (1)-(5) show the meta-alert generation procedure using alerts from system A. Steps (6) indicates the detection of a similar alert group on system B. 153
7.6 Scenario for cross-system alert merging. Steps (1)-(8) alternate between knowledge base extensions and repeated meta-alert generation. 155
7.7 Alerts from AMiner (1) and Wazuh (2) IDS on four web servers. From left to right, the attacks are Nmap scans (red), Nikto vulnerability scans (blue), user enumerations (green), Hydra brute-force login attempts (pink), webmail exploits for webshell uploads (orange), and Exim exploits for privilege escalation (purple). 164
7.8 Influence of δ value on the number of generated alert groups on four systems. 165
7.9 Multi-dimensional scaling of pairwise group similarities using δ values of 10 seconds (left) and 1 second (right) shows that groups of same attack steps are frequently located close together. 166
7.10 Dendrogram of group similarities, where each node represents a merged group. The dashed line shows a reasonable cutoff threshold at a similarity of 0.3 that yields meta-alerts with few incorrect allocations. 167
7.11 Cluster purity and the total number of clusters suggests 0.3 as a similarity cutoff threshold. 168
7.12 Reduction rate plotted against false positive rate for several threshold values. 168
7.13 Left: True positive rate plotted against false positive rate of cross-system attack classifications for several threshold values. Right: Confusion matrix of attack classifications using 0.3 as threshold. 170
7.14 F1-score boxplots of cross-system classification for several δ values, categorized by attack and θ_{group}. 171
7.15 Influence of δ and θ_{group} on reduction rates. Left: Group reduction rates. Right: Alert reduction rates. 172
7.16 Robustness to randomly generated noise alerts for $\delta = 5$ seconds and $\theta_{group} = \theta_{alert} = 0.3$. Left: Influence on number of alert groups. Right: Influence on TPR, FPR, and F1-score. 173
7.17 Runtime required to process alerts for different values for different δ (left) and thresholds (right). 174

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

List of Tables

2.1 Fulfillment of requirements for existing datasets 15
2.2 Fulfillment of requirements for existing alert aggregation approaches . . . 26

3.1 Testbed development phases 32

4.1 Variations of the system environment 57
4.2 Variations of simulated user behavior 59
4.3 Overview of the attack scenario 63
4.4 Variations of the attack scenario 64
4.5 Technical infrastructure of testbeds 71
4.6 Log files collected from hosts 72
4.7 Overview of the number of rule templates for each log data label 78
4.8 Overview of the number of labeled lines per file for each log data label . . 79

5.1 Overview of main goals of reviewed approaches (categorizations are not mutually exclusive) 100
5.2 Assessed properties regarding clustering techniques assigned to each approach 102
5.3 Assessed properties regarding the evaluation carried out in each approach . 111

6.1 Value co-occurrences of syscall types and items in Audit logs 119
6.2 Definitions of symbols used in this chapter 120
6.3 Sample data 122
6.4 Dependencies and default values of thresholds 134

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Bibliography

[AAAH+18] Hamad Almohannadi, Irfan Awan, Jassim Al Hamar, Andrea Cullen, Jules Pagan Disso, and Lorna Armitage. Cyber threat intelligence from honeypot data using elasticsearch. In *Proceedings of the 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 900–906. IEEE, 2018.

[ABCM09] Michal Aharon, Gilad Barash, Ira Cohen, and Eli Mordechai. One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 227–243. Springer, 2009.

[ACM] ACM Digital Library. `https://dl.acm.org/results.cfm`. Accessed: 2021-10-18.

[AEC] Aecid website. `https://aecid.ait.ac.at/`. Accessed: 2021-10-18.

[AGR+19] Jawad Ahmed, Hassan Habibi Gharakheili, Qasim Raza, Craig Russell, and Vijay Sivaraman. Monitoring enterprise DNS queries for detecting data exfiltration from internal hosts. *IEEE Transactions on Network and Service Management*, 17(1):265–279, 2019.

[AHH20] Yahya Al-Hadhrami and Farookh Khadeer Hussain. Real time dataset generation framework for intrusion detection systems in IoT. *Future Generation Computer Systems*, 108:414–423, 2020.

[AKBS19] Aman Arora, Anureet Kaur, Bharat Bhushan, and Himanshu Saini. Security concerns and future trends of internet of things. In *Proceedings of the 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 891–896. IEEE, 2019.

[AKG19] Amey Agrawal, Rohit Karlupia, and Rajat Gupta. Logan: A distributed online log parser. In *Proceedings of the 35th International Conference on Data Engineering (ICDE)*, pages 1946–1951. IEEE, 2019.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Ale] Aecid alert aggregation. https://github.com/ait-aecid/aecid-alert-aggregation. Accessed: 2021-10-18.

[Als16] Faeiz Alserhani. Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack. *International Journal of Advanced Studies in Computers, Science and Engineering*, 5(2):1, 2016.

[AMi] Logdata anomaly miner (aminer). https://github.com/ait-aecid/logdata-anomaly-miner. Accessed: 2021-10-18.

[AMZ09] Safaa Al-Mamory and Hongli Zhang. Intrusion detection alarms reduction using root cause analysis and clustering. *Computer Communications*, 32(2):419–430, 2009.

[Ans] Red Hat Ansible Automation Platform. https://www.ansible.com/. Accessed: 2021-10-18.

[APC18] Nicolas Aussel, Yohan Petetin, and Sophie Chabridon. Improving performances of log mining for anomaly prediction through NLP-based log parsing. In *Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 237–243. IEEE, 2018.

[AR19] Rachel Allen and Bartley Richardson. Neural network, that's the tech; to free your staff from, bad regex. https://medium.com/rapids-ai/cybert-28b35a4c81c4, December 2019. Accessed 2021-11-26.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499. Citeseer, 1994.

[ASZ+16] Taqwa Ahmed Alhaj, Maheyzah Md Siraj, Anazida Zainal, Huwaida Tagelsir Elshoush, and Fatin Elhaj. Feature selection using information gain for improved structural-based alert correlation. *PloS one*, 11:1–18, 2016.

[Bar06] Sean Barnum. Common attack pattern enumeration and classification (CAPEC) schema description. Technical report, 2006. Cigital.

[Bar12] Sean Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (STIX). Technical report, 2012. Mitre Corporation.

[BB00] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining*, pages 407–416. ACM, 2000.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[BBG13] Mehdi Bateni, Ahmad Baraani, and Ali Ghorbani. Using artificial immune system and fuzzy logic for alert correlation. *International Journal of Network Security*, 15(3):190–204, 2013.

[Ber13] Wicher Bergsma. A bias-correction for Cramér's V and Tschuprow's T. *Journal of the Korean Statistical Society*, 42(3):323–328, 2013.

[BG05] Ingwer Borg and Patrick Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[BHO+20] Erik Brynjolfsson, John J Horton, Adam Ozimek, Daniel Rock, Garima Sharma, and Hong-Yi TuYe. Covid-19 and remote work: an early look at US data. Technical report, National Bureau of Economic Research, 2020.

[Bia14] David Bianco. The pyramid of pain: Intel-driven detection & response to increase your adversary's cost of operations. Technical report, 2014. RVASec.

[Bip] ggbiplot R Package Github Repository. `https://github.com/vqv/ggbiplot`. Accessed: 2021-10-18.

[BJS+11] Sorana D Bolboacă, Lorentz Jäntschi, Adriana F Sestraş, Radu E Sestraş, and Doru C Pamfil. Pearson-fisher chi-square statistic revisited. *Information*, 2(3):528–545, 2011.

[BLL+18] Liang Bao, Qian Li, Peiyao Lu, Jie Lu, Tongxiao Ruan, and Ke Zhang. Execution anomaly detection in large-scale systems through console log analysis. *Journal of Systems and Software*, 143:172–186, 2018.

[BS06] Raju Balakrishnan and Ramendra K Sahoo. Lossless compression for large scale cluster logs. In *Proceedings of the 20th International Parallel & Distributed Processing Symposium*, page 7. IEEE, 2006.

[CAPa] Common Attack Pattern Enumeration and Classification (CAPEC). `https://capec.mitre.org/`. Accessed: 2021-10-18.

[Capb] Capri Dataset. `https://research.cs.queensu.ca/home/farhana/supporting-pages/capri.html`. Accessed: 2021-10-18.

[Car12] David Carasso. *Exploring splunk*. CITO Research, 2012.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15:1–15:58, 2009.

[CFD] CFDR Data. `https://www.usenix.org/cfdr-data`. Accessed: 2021-10-18.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[ČG18] Dainius Čeponis and Nikolaj Goranin. Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset. *Baltic Journal of Modern Computing*, 6(3):217–234, 2018.

[CH13] Gideon Creech and Jiankun Hu. Generation of a new IDS test dataset: Time to retire the KDD collection. In *Proceedings of the Wireless Communications and Networking Conference*, pages 4487–4492. IEEE, 2013.

[Chr] Chromium Browser. `https://www.chromium.org/`. Accessed: 2021-10-18.

[Chu10] Anton Chuvakin. Public security log sharing site. `https://log-sharing.dreamhosters.com/`, November 2010. Accessed: 2021-10-18.

[CKH+10] Edward Chuah, Shyh-hao Kuo, Paul Hiew, William-Chandra Tjhi, Gary Lee, John Hammond, Marek T Michalewicz, Terence Hung, and James C Browne. Diagnosing the root-causes of failures from cluster log files. In *Proceedings of the International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE, 2010.

[CL13] Robert Christensen and Feifei Li. Adaptive log compression for massive log data. In *Proceedings of the International Conference on Management of Data*, page 1283. ACM, 2013.

[Cla] ClamAV Open-Source Antivirus. `https://www.clamav.net/`. Accessed: 2021-10-18.

[CM02] Frédéric Cuppens and Alexandre Miege. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the Symposium on Security and Privacy*, pages 202–215. IEEE, 2002.

[CORW09] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17:1–17:38, 2009.

[CR15] David Chismon and Martyn Ruks. Threat intelligence: Collecting, analysing, evaluating. Technical report, 2015. MWR InfoSecurity.

[Cre14] Gideon Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.* PhD thesis, University of New South Wales, Canberra, Australia, 2014.

[Cro21] Crowdstrike global threat report. `https://www.crowdstrike.com/resources/reports/global-threat-report/`, 2021. Accessed: 2021-11-23.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[CTS+16] Ting Chen, Lu-An Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. Entity embedding-based anomaly detection for heterogeneous categorical events. *arXiv preprint arXiv:1608.07502*, 2016.

[DABJM+18] Sean Carlisto De Alvarenga, Sylvio Barbon Jr, Rodrigo Sanches Miani, Michel Cukier, and Bruno Bogaz Zarpelão. Process mining and hierarchical clustering to help intrusion alert visualization. *Computers & Security*, 73:474–491, 2018.

[Dal14] Henry Dalziel. *How to define and build an effective cyber threat intelligence capability.* Syngress, 2014.

[DBFV18] Youcef Djenouri, Asma Belhadi, and Philippe Fournier-Viger. Extracting useful knowledge from event logs: a frequent itemset mining approach. *Knowledge-Based Systems*, 139:132–148, 2018.

[DC15] Sizhong Du and Jian Cao. Behavioral anomaly detection approach based on log monitoring. In *Proceedings of the International Conference on Behavioral, Economic and Socio-cultural Computing (BESC)*, pages 188–194. IEEE, 2015.

[Dir] Dirb Web Scanner. `http://dirb.sourceforge.net/`. Accessed: 2021-10-18.

[DL16] Min Du and Feifei Li. Spell: Streaming parsing of system event logs. In *Proceedings of the 16th International Conference on Data Mining (ICDM)*, pages 859–864. IEEE, 2016.

[DLZS17] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the Conference on Computer and Communications Security*, pages 1285–1298. ACM, 2017.

[Dnsa] Dnsmasq 2.79. `https://thekelleys.org.uk/dnsmasq/doc.html`. Accessed: 2021-10-18.

[DNSb] DNSteal data extraction through DNS requests. `https://github.com/m57/dnsteal`. Accessed: 2021-10-18.

[DS07] Kaustav Das and Jeff Schneider. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, pages 220–229. ACM, 2007.

[E+13] Vinodh Ewards et al. A survey on signature generation methods for network traffic classification. *International Journal of Advanced Research in Computer Science*, 4(2), 2013.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[EFMRGB+19] Carlos Eiras-Franco, David Martinez-Rego, Bertha Guijarro-Berdinas, Amparo Alonso-Betanzos, and Antonio Bahamonde. Large scale anomaly detection in mixed numerical and categorical input spaces. *Information Sciences*, 487:115–127, 2019.

[Ela] Elasticsearch. `https://www.elastic.co/`. Accessed: 2021-10-18.

[ELLS11] Brian S Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. *Cluster analysis 5th ed.* John Wiley, 2011.

[ENI21] Enisa threat landscape. `https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021`, 2021. Accessed: 2021-11-23.

[EO11] Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman. Alert correlation in collaborative intelligent intrusion detection systems - a survey. *Applied Soft Computing*, 11(7):4349–4365, 2011.

[EQL] Event Query Language. `https://www.elastic.co/guide/en/elasticsearch/reference/current/eql.html`. Accessed: 2021-10-18.

[Fak] Faker python library. `https://github.com/joke2k/faker`. Accessed: 2021-10-18.

[FHSL96] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. A sense of self for unix processes. In *Proceedings of the Symposium on Security and Privacy*, pages 120–128. IEEE, 1996.

[Fira] M-trends 2010: The advanced persistent threat. `https://www.fireeye.com/current-threats/annual-threat-report/mtrends/rpt-2010-mtrends.html`. Accessed: 2021-10-18.

[Firb] Firefox Browser. `https://www.mozilla.org/en-US/firefox/`. Accessed: 2021-10-18.

[FL05] Federico Michele Facca and Pier Luca Lanzi. Mining interesting knowledge from weblogs: a survey. *Data & Knowledge Engineering*, 53(3):225–241, 2005.

[FLWL09] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. Execution anomaly detection in distributed systems through unstructured log analysis. In *Proceedings of the 9th International Conference on Data Mining (ICDM'09)*, pages 149–158. IEEE, 2009.

[Fra21] Maximilian Frank. Quality improvement of labels for model-driven benchmark data generation for intrusion detection systems, 2021. Master's Thesis, Vienna University of Technology.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[GCH+11] Georges Grinstein, Kristin Cook, Paul Havig, Kristen Liggett, Bohdan Nebesh, Mark Whiting, Kirsten Whitley, and Shawn Konecni. VAST Challenge 2011 MC2 - Computer Networking Operations. http://visualdata.wustl.edu/varepository/VAST%20Challenge%202011/challenges/MC2%20-%20Computer%20Networking%20Operations/, May 2011. Accessed: 2021-10-18.

[GCTMK11] Ana Gainaru, Franck Cappello, Stefan Trausan-Matu, and Bill Kramer. Event log mining tool for large scale HPC systems. In *Proceedings of the European Conference on Parallel Processing*, pages 52–64. Springer, 2011.

[GdVFM08] Fermin Galan, Jorge E Lopez de Vergara, David Fernandez, and Raul Munoz. A model-driven configuration management methodology for testbed infrastructures. In *Proceedings of the Network Operations and Management Symposium*, pages 747–750. IEEE, 2008.

[GFDVC10] Fermin Galan, David Fernandez, Jorge E López De Vergara, and Ramon Casellas. Using a model-driven architecture for technology-independent scenario configuration in networking testbeds. *IEEE Communications Magazine*, 48(12):132–141, 2010.

[Gho20] Morteza Ghobakhloo. Industry 4.0, digitization, and opportunities for sustainability. *Journal of Cleaner Production*, 252:119869, 2020.

[GJL+15] Nentawe Gurumdimma, Arshad Jhumka, Maria Liakata, Edward Chuah, and James Browne. Towards detecting patterns in failure logs of large-scale distributed systems. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, pages 1052–1061. IEEE, 2015.

[GK16] Govind P Gupta and Manish Kulariya. A framework for fast and efficient cyber security network intrusion detection using apache spark. *Procedia Computer Science*, 93:824–831, 2016.

[Goo] Google Scholar. https://scholar.google.at/. Accessed: 2021-10-18.

[GRKG19] Martin Grimmer, Martin Max Röhling, D Kreusel, and Simon Ganz. A modern and sophisticated host based intrusion detection data set. *IT-Sicherheit als Voraussetzung für eine erfolgreiche Digitalisierung*, pages 135–145, 2019.

[GSL+20] Peng Gao, Fei Shao, Xiaoyuan Liu, Xusheng Xiao, Zheng Qin, Fengyuan Xu, Prateek Mittal, Sanjeev R Kulkarni, and Dawn Song. Enabling efficient cyber threat hunting with cyber threat intelligence. *arXiv preprint arXiv:2010.13637*, 2020.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[HASA+17] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 103–115. ACM, 2017.

[HČLV17] Martin Husák, Milan Čermák, Martin Laštovička, and Jan Vykopal. Exchanging security events: Which and how many alerts can we aggregate? In *Proceedings of the Symposium on Integrated Network and Service Management*, pages 604–607. IEEE, 2017.

[HDX+16] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th International Conference on Information and Knowledge Management*, pages 1573–1582. ACM, 2016.

[HHC+21] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. A survey on automated log analysis for reliability engineering. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.

[HHS+17] Waqas Haider, Jiankun Hu, Jill Slay, Benjamin P Turnbull, and Yi Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, 2017.

[HK19] Martin Husák and Jaroslav Kašpar. AIDA framework: Real-time correlation and prediction of intrusion detection alerts. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–8. ACM, 2019.

[HLF+20] Shaohan Huang, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. Transfer log-based anomaly detection with pseudo labels. In *Proceedings of the 16th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2020.

[Hor] Horde Groupware 5.2.17. `https://www.horde.org/apps/groupware`. Accessed: 2021-10-18.

[HS09] Alexander Hofmann and Bernhard Sick. Online intrusion alert aggregation with generative data stream modeling. *IEEE Transactions on Dependable and Secure Computing*, 8(2):282–294, 2009.

[HWF19] Steffen Haas, Florian Wilkens, and Mathias Fischer. Efficient attack correlation and identification of attack scenarios based on network-motifs. In *Proceedings of the 38th International Performance Computing and Communications Conference*, pages 1–11. IEEE, 2019.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Hyd] Hydra Tool. `https://tools.kali.org/password-attacks/hydra`. Accessed: 2021-10-18.

[HZH+17] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R Lyu. Towards automated log parsing for large-scale log data analysis. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[HZHL16] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Experience report: System log analysis for anomaly detection. In *Proceedings of the 27th International Symposium on Software Reliability Engineering*, pages 207–218. IEEE, 2016.

[HZHL20] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448*, 2020.

[HZZL17] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. Drain: An online log parsing approach with fixed depth tree. In *Proceedings of the International Conference on Web Services (ICWS)*, pages 33–40. IEEE, 2017.

[IDM] RFC4765: The Intrusion Detection Message Exchange Format (IDMEF). `https://datatracker.ietf.org/doc/html/rfc4765`. Accessed: 2021-10-18.

[IEE] IEEE Xplore Digital Library. `https://ieeexplore.ieee.org/Xplore/home.jsp`. Accessed: 2021-10-18.

[IOD] RFC5070: The Incident Object Description Exchange Format (IODEF). `https://datatracker.ietf.org/doc/html/rfc5070`. Accessed: 2021-10-18.

[IPM16] Dino Ienco, Ruggero G Pensa, and Rosa Meo. A semisupervised approach to the detection and characterization of outliers in categorical data. *IEEE Transactions on Neural Networks and Learning Systems*, 28(5):1017–1029, 2016.

[ISO10] Information technology - Security techniques - Network security - Part 3: Reference networking scenarios - Threats, design techniques and control issues, December 2010.

[JBG14] Basanta Joshi, Umanga Bista, and Manoj Ghimire. Intelligent clustering scheme for log data streams. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics*, pages 454–465. Springer, 2014.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[JHHF08] Zhen Ming Jiang, Ahmed E Hassan, Gilbert Hamann, and Parminder Flora. An automated approach for abstracting execution logs to execution events. *Journal of Software: Evolution and Process*, 20(4):249–267, 2008.

[Jin] Jinja Templating Engine. `https://jinja.palletsprojects.com/en/3.0.x/`. Accessed: 2021-10-18.

[Joh] John the Ripper Password Cracker. `https://www.openwall.com/john/`. Accessed: 2021-10-18.

[JSC+09] Sourabh Jain, Inderpreet Singh, Abhishek Chandra, Zhi-Li Zhang, and Greg Bronevetsky. Extracting the textual and temporal structure of supercomputing logs. In *Proceedings of the International Conference on High Performance Computing (HiPC)*, pages 254–263. IEEE, 2009.

[JSH15] Antti Juvonen, Tuomo Sipola, and Timo Hämäläinen. Online anomaly detection using dimensionality reduction techniques for HTTP log analysis. *Computer Networks*, 91:46–56, 2015.

[Jul03] Klaus Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6(4):443–471, 2003.

[JVH+19] Jiaojiao Jiang, Steve Versteeg, Jun Han, Md Arafat Hossain, Jean-Guy Schneider, Christopher Leckie, and Zeinab Farahmandpour. P-gram: Positional n-gram for the clustering of machine-generated messages. *IEEE Access*, 7:88504–88516, 2019.

[JWH17] PWDC Jayathilake, NR Weeraddana, and HKEP Hettiarachchi. Automatic detection of multi-line templates in software log files. In *Proceedings of the 17th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 1–8. IEEE, 2017.

[JYC+17] Tong Jia, Lin Yang, Pengfei Chen, Ying Li, Fanjing Meng, and Jingmin Xu. Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs. In *Proceedings of the 10th International Conference on Cloud Computing (CLOUD)*, pages 447–455. IEEE, 2017.

[Kal] Kali Penetration Testing Complete Tools List. `https://en.kali.tools/all/`. Accessed: 2021-10-18.

[KAW11] Md Tanzim Khorshed, ABM Shawkat Ali, and Saleh A Wasimi. Monitoring insiders activities in cloud computing using rule based learning. In *Proceedings of the 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 757–764. IEEE, 2011.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[KB12] Hisham A Kholidy and Fabrizio Baiardi. CIDD: A cloud intrusion detection dataset for cloud computing and masquerade attacks. In *Proceedings of the 9th International Conference on Information Technology-New Generations*, pages 397–402. IEEE, 2012.

[KFE14] Satoru Kobayashi, Kensuke Fukuda, and Hiroshi Esaki. Towards an NLP-based log template generation algorithm for system log analysis. In *Proceedings of the 9th International Conference on Future Internet Technologies*, pages 11:1–11:4. ACM, 2014.

[KGVK19] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[KIM+14] Tatsuaki Kimura, Keisuke Ishibashi, Tatsuya Mori, Hiroshi Sawada, Tsuyoshi Toyono, Ken Nishimatsu, Akio Watanabe, Akihiro Shimoda, and Kohei Shiomoto. Spatio-temporal factorization of log data for understanding network events. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, pages 610–618. IEEE, 2014.

[KKA14] Issa M Khalil, Abdallah Khreishah, and Muhammad Azeem. Cloud computing security: A survey. *Computers*, 3(1):1–35, 2014.

[KS15] Abdullah Khalili and Ashkan Sami. SysDetect: a systematic approach to critical state determination for industrial intrusion detection systems using apriori algorithm. *Journal of Process Control*, 32:154–160, 2015.

[KV03] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th conference on Computer and communications security*, pages 251–261. ACM, 2003.

[Kyo] Kyoushi testbed. `https://github.com/ait-aecid/kyoushi-environment`. Accessed: 2021-10-18.

[LCWX16] Wei Liang, Zuo Chen, Ya Wen, and Weidong Xiao. An alert fusion method based on grey relation and attribute similarity correlation. *International Journal of Online and Biomedical Engineering*, 12(08):25–30, 2016.

[LDF+18] Zongze Li, Matthew Davidson, Song Fu, Sean Blanchard, and Michael Lang. Converting unstructured system logs into structured event list for anomaly detection. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 15:1–15:10. ACM, 2018.

[LFS+21] Max Landauer, Maximilian Frank, Florian Skopik, Wolfgang Hotwagner, Markus Wurzenberger, and Andreas Rauber. Kyoushi Log Data Set. `https://doi.org/10.5281/zenodo.5779411`, December 2021.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[LFS+22] Max Landauer, Maximilian Frank, Florian Skopik, Wolfgang Hotwagner, Markus Wurzenberger, and Andreas Rauber. A framework for automatic labeling of log datasets from model-driven testbeds for HIDS evaluation. In *Proceedings of the Workshop on Secure and Trustworthy Cyber-Physical Systems*. ACM, 2022. Accepted for publication.

[LHW+21] Max Landauer, Georg Höld, Markus Wurzenberger, Florian Skopik, and Andreas Rauber. Iterative selection of categorical variables for log data anomaly detection. In *Proceedings of the European Symposium on Research in Computer Security*, pages 757–777. Springer, 2021.

[LJZ+17] Tao Li, Yexi Jiang, Chunqiu Zeng, Bin Xia, Zheng Liu, Wubai Zhou, Xiaolong Zhu, Wentao Wang, Liang Zhang, Jun Wu, et al. Flap: An end-to-end event log analysis platform for system management. In *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining*, pages 1547–1556. ACM, 2017.

[LLLT13] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[LLMP05] Tao Li, Feng Liang, Sheng Ma, and Wei Peng. An integrated framework on mining logs files for computing system management. In *Proceedings of the 11th International Conference on Knowledge Discovery in Data Mining*, pages 776–781. ACM, 2005.

[Log] Logstash. https://www.elastic.co/de/logstash/. Accessed: 2021-10-18.

[LPC+13] Beatriz Pérez Lamancha, Macario Polo, Danilo Caivano, Mario Piattini, and Giuseppe Visaggio. Automated generation of test oracles using a model-driven approach. *Information and Software Technology*, 55(2):301–319, 2013.

[LS19] Max Landauer and Florian Skopik. INDICÆTING–automatically detecting, extracting, and correlating cyber threat intelligence from raw computer log data. *ERCIM NEWS*, (116):25–26, 2019.

[LSF+] Max Landauer, Florian Skopik, Maximilian Frank, Wolfgang Hotwagner, Markus Wurzenberger, and Andreas Rauber. Maintainable log datasets for evaluation of intrusion detection systems. Under review.

[LSF+21] Max Landauer, Florian Skopik, Maximilian Frank, Wolfgang Hotwagner, Markus Wurzenberger, and Andreas Rauber. AIT Log Data Set V2.0. https://doi.org/10.5281/zenodo.5789063, 2021.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[LSS06] Jidong Long, Daniel Schwartz, and Sara Stoecklin. Distinguishing false from true alerts in snort by data mining patterns of alerts. In *Proceedings of the Defense and Security Symposium*, volume 6241, pages 99–108. International Society for Optics and Photonics, 2006.

[LSW+19] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. A framework for cyber threat intelligence extraction from raw log data. In *Proceedings of the International Conference on Big Data*, pages 3200–3209. IEEE, 2019.

[LSW+20a] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. AIT Log Data Set V1.1. `https://doi.org/10.5281/zenodo.4264796`, November 2020.

[LSW+20b] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. Visualizing syscalls using self-organizing maps for system intrusion detection. In *Proceedings of the International Conference on Information Systems Security and Privacy*, pages 349–360. SCITEPRESS, 2020.

[LSW+21] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. Have it your way: Generating customized log datasets with a model-driven simulation testbed. *IEEE Transactions on Reliability*, 70(1):402–415, 2021.

[LSWR20] Max Landauer, Florian Skopik, Markus Wurzenberger, and Andreas Rauber. System log clustering approaches for cyber security applications: A survey. *Computers & Security*, 92:101739, 2020.

[LSWR22] Max Landauer, Florian Skopik, Markus Wurzenberger, and Andreas Rauber. Dealing with security alert flooding: Using machine learning for domain-independent alert aggregation. *ACM Transactions on Privacy and Security*, 2022. Accepted for publication.

[LTPM17] Laetitia Leichtnam, Eric Totel, Nicolas Prigent, and Ludovic Mé. STARLORD: Linked security data exploration in a 3D graph. In *Proceedings of the Symposium on Visualization for Cyber Security (VizSec)*, pages 1–4. IEEE, 2017.

[LWS+18a] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. *Computers & Security*, 79:94–116, 2018.

[LWS+18b] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. Time series analysis: unsupervised anomaly detection beyond outlier detection. In *Proceedings of the International*

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Conference on Information Security Practice and Experience, pages 19–36. Springer, 2018.

[LZL+16] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 102–111. ACM, 2016.

[Mar] MaraDNS 2.0.13. `https://maradns.samiam.org/`. Accessed: 2021-10-18.

[MC17] Ramiro Montealegre and Wayne F Cascio. Technology-driven changes in work and employment. *Communications of the ACM*, 60(12):60–67, 2017.

[McM13] Rob McMillan. Definition: Threat intelligence. `https://www.gartner.com/en/documents/2487216`, 2013. Accessed: 2021-12-10.

[Met] Metasploit. `https://www.metasploit.com/`. Accessed: 2021-10-18.

[MFCMC+18] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Theron. UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security*, 73:411–424, 2018.

[MIT] Mitre att&ck matrix. `https://attack.mitre.org/`. Accessed: 2021-10-18.

[Miz13] Masayoshi Mizutani. Incremental mining of system log format. In *Proceedings of the International Conference on Services Computing (SCC)*, pages 595–602. IEEE, 2013.

[MP17] Vlado Menkovski and Milan Petkovic. Towards unsupervised signature extraction of forensic logs. In *Proceedings of the 26th Benelux Conference on Machine Learning*, pages 154–160. Springer, 2017.

[MPB+18] Salma Messaoudi, Annibale Panichella, Domenico Bianculli, Lionel Briand, and Raimondas Sasnauskas. A search-based approach for accurate identification of log message formats. In *Proceedings of the 26th International Conference on Program Comprehension (ICPC'18)*. ACM, 2018.

[MS16] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[MSR08] Christopher Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge University Press, 2008.

[MYK18] Stephen Moskal, Shanchieh Jay Yang, and Michael Kuhl. Extracting and evaluating similar and unique cyber attack strategies from intrusion alerts. In *Proceedings of the International Conference on Intelligence and Security Informatics*, pages 49–54. IEEE, 2018.

[MYWX12] Dapeng Man, Wu Yang, Wei Wang, and Shichang Xuan. An alert aggregation algorithm based on iterative self-organization. *Procedia Engineering*, 29:3033–3038, 2012.

[MZHM09a] Adetokunbo Makanju, A Nur Zincir-Heywood, and Evangelos E Milios. Clustering event logs using iterative partitioning. In *Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining*, pages 1255–1264. ACM, 2009.

[MZHM+09b] Adetokunbo Makanju, A Nur Zincir-Heywood, Evangelos E Milios, et al. Extracting message types from BlueGene/L's logs. In *Proceedings of the SOSP Workshop on the Analysis of System Logs (WASL)*. ACM, 2009.

[N+17] Julio Navarro et al. Huma: A multi-layer framework for threat analysis in a heterogeneous log environment. In *Proceedings of the 10th International Symposium on Foundations and Practice of Security*, pages 144–159. Springer, 2017.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[NCR02] Peng Ning, Yun Cui, and Douglas Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th Conference on Computer and Communications Security*, pages 245–254. ACM, 2002.

[NDP16] Julio Navarro, Aline Deruyver, and Pierre Parrend. Morwilog: an ACO-based system for outlining multi-step attacks. In *Proceedings of the Symposium Series on Computational Intelligence*, pages 1–8. IEEE, 2016.

[NDP18] Julio Navarro, Aline Deruyver, and Pierre Parrend. A systematic survey on multi-step attack detection. *Computers & Security*, 76:214–249, 2018.

[Nik] Nikto Vulnerability Scanning Tool. `https://cirt.net/Nikto2`. Accessed: 2021-10-18.

[NJCY14] Xia Ning, Geoff Jiang, Haifeng Chen, and Kenji Yoshihira. HLAer: a system for heterogeneous log analysis. In *Proceedings of the SDM Workshop on Heterogeneous Learning*. Citeseer, 2014.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[NK08] Kazuyo Narita and Hiroyuki Kitagawa. Detecting outliers in categorical record databases based on attribute associations. In *Proceedings of the Asia-Pacific Web Conference*, pages 111–123. Springer, 2008.

[NK19] Petteri Nevavuori and Tero Kokkonen. Requirements for training and evaluation dataset of network and host intrusion detection system. In *Proceedings of the World Conference on Information Systems and Technologies*, pages 534–546. Springer, 2019.

[Nma] Nmap Port Scanning Tool. `https://nmap.org/`. Accessed: 2021-10-18.

[NMA+16] Animesh Nandi, Atri Mandal, Shubham Atreja, Gargi B Dasgupta, and Subhrajit Bhattacharya. Anomaly detection using program control flow graph mining from execution logs. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 215–224. ACM, 2016.

[NSC16] John Narayan, Sandeep K Shukla, and T Charles Clancy. A survey of automatic protocol reverse engineering tools. *ACM Computing Surveys (CSUR)*, 48(3):40:1–40:26, 2016.

[NV10] Meiyappan Nagappan and Mladen A Vouk. Abstracting log lines to log event types for mining software system logs. In *Proceedings of the 7th Working Conference on Mining Software Repositories (MSR)*, pages 114–117. IEEE, 2010.

[Oka] OkayCMS. `https://okay-cms.com/`. Accessed: 2021-10-18.

[Opea] OpenIOC. `https://github.com/fireeye/OpenIOC_1.1`. Accessed: 2021-10-18.

[Opeb] OpenStack Open Source Cloud Software. `https://www.openstack.org/`. Accessed: 2021-10-18.

[Opec] OpenVPN 2.4.4. `https://openvpn.net/`. Accessed: 2021-10-18.

[Own] OwnCloud 10.5.0. `https://owncloud.com/`. Accessed: 2021-10-18.

[PA17] Amit Pande and Vishal Ahuja. WEAC: Word embeddings for anomaly classification from event logs. In *Proceedings of the International Conference on Big Data*, pages 1095–1100. IEEE, 2017.

[PBS+11] Robert Patton, Justin Beaver, Chad Steed, Thomas Potok, and Jim Treadwell. Hierarchical clustering and visualization of aggregate cyber data. In *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference*, pages 1287–1291. IEEE, 2011.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[PES01] Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA)*, pages 5–8. Citeseer, 2001.

[PGS+16] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. HERCULE: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 583–595. ACM, 2016.

[Pla21] KSÖ Planspiel, Austrian Press Agency. `https://www.ots.at/presseaussendung/OTS_20210922_OTS0036`, September 2021. Accessed: 2021-10-18.

[PMGO18] Souneil Park, Aleksandar Matic, Kamini Garg, and Nuria Oliver. When simpler data does not imply less information: A study of user profiling scenarios with constrained view of mobile HTTP(S) traffic. *ACM Transactions on the Web (TWEB)*, 12(2):1–23, 2018.

[PRT+18] Davy Preuveneers, Vera Rimmer, Ilias Tsingenopoulos, Jan Spooren, Wouter Joosen, and Elisabeth Ilie-Zudor. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 8(12):2663, 2018.

[PTVF07] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[QGP+10] Tongqing Qiu, Zihui Ge, Dan Pei, Jia Wang, and Jun Xu. What happened in my network: Mining network events from router syslogs. In *Proceedings of the 10th Conference on Internet Measurement*, pages 472–484. ACM, 2010.

[QZLS12] Lin-Bo Qiao, Bo-Feng Zhang, Zhi-Quan Lai, and Jin-Shu Su. Mining of attack models in IDS alerts from network backbone by a two-stage clustering method. In *Proceedings of the 26th International Parallel and Distributed Processing Symposium*, pages 1263–1269. IEEE, 2012.

[RAA15] Ali Ahmadian Ramaki, Morteza Amini, and Reza Ebrahimi Atani. Rteca: Real time episode correlation algorithm for multi-step attack scenarios detection. *Computers & Security*, 49:206–219, 2015.

[RCY+18] Rui Ren, Jiechao Cheng, Yan Yin, Jianfeng Zhan, Lei Wang, Jinheng Li, and Chunjie Luo. Deep convolutional neural networks for log event classification on distributed cluster systems. In *Proceedings of the International Conference on Big Data*, pages 1639–1646. IEEE, 2018.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[Rho14] Doug Rhoades. Machine actionable indicators of compromise. In *Proceedings of the 48th International Carnahan Conference on Security Technology*, pages 1–5. IEEE, 2014.

[RJW11] Thomas Reidemeister, Miao Jiang, and Paul AS Ward. Mining unstructured log files for recurrent fault diagnosis. In *Proceedings of the International Symposium on Integrated Network Management (IM)*, pages 377–384. IEEE, 2011.

[Rou87] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[RSG10] Hanli Ren, Natalia Stakhanova, and Ali Ghorbani. An online adaptive approach to alert correlation. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 153–172. Springer, 2010.

[RWG+17] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security*, pages 361–369. ACPIL, 2017.

[RWZH09] Jiadong Ren, Qunhui Wu, Jia Zhang, and Changzhen Hu. Efficient outlier detection algorithm for heterogeneous data streams. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery*, volume 5, pages 259–264. IEEE, 2009.

[Sam] Samba 4.5.9. `https://samba.org/`. Accessed: 2021-10-18.

[SAvD19] Daan Schipper, Maurício Aniche, and Arie van Deursen. Tracing back log data to its log statement: from research to practice. In *Proceedings of the 16th International Conference on Mining Software Repositories*, pages 545–549. IEEE Press, 2019.

[SBJ+20] Mustafizur R Shahid, Gregory Blanc, Houda Jmila, Zonghua Zhang, and Hervé Debar. Generative deep learning for internet of things network traffic generation. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 70–79. IEEE, 2020.

[SBL+09] Hassan Saneifar, Stéphane Bonniol, Anne Laurent, Pascal Poncelet, and Mathieu Roche. Terminology extraction from log files. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 769–776. Springer, 2009.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[Sch06] Douglas C Schmidt. Model-driven engineering. *IEEE Computer Society*, 39(2):25, 2006.

[SE19] Moza Shibani and Anupriya E. Automated threat hunting using ELK stack - a case study. *Indian Journal of Computer Science and Engineering*, 10:118–127, 10 2019.

[Sel] Selenium Web Automation Software. `https://www.selenium.dev/`. Accessed: 2021-10-18.

[SFK15] Nicandro Scarabeo, Benjamin CM Fung, and Rashid H Khokhar. Mining known attack patterns from security-related events. *PeerJ Computer Science*, 1:21, 2015.

[SFL+99] J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. KDD cup 1999 data. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, October 1999. Accessed: 2021-10-18.

[SG09] Reza Sadoddin and Ali Ghorbani. An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security*, 28(3-4):153–173, 2009.

[SG+20] Jiaxuan Sun, Lize Gu, et al. An efficient alert aggregation method based on conditional rough entropy and knowledge granularity. *Entropy*, 22(3):1–23, 2020.

[SGIM21] John Shier, Mat Gangwer, Greg Iddon, and Peter Mackenzie. The active adversary playbook. `https://news.sophos.com/en-us/2021/05/18/the-active-adversary-playbook-2021/`, 2021. Accessed: 2021-11-23.

[SHGH+15] Riyanat Shittu, Alex Healing, Robert Ghanea-Hercock, Robin Bloomfield, and Muttukrishnan Rajarajan. Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1–15, 2015.

[SHH+06] Salvatore J Stolfo, Shlomo Hershkop, Chia-Wei Hu, Wei-Jen Li, Olivier Nimeskern, and Ke Wang. Behavior-based modeling and its application to email analysis. *ACM Transactions on Internet Technology*, 6(2):187–221, 2006.

[Shi16] Keiichi Shima. Length matters: Clustering system log messages using length of words. *arXiv preprint arXiv:1611.03213*, 2016.

[Sho] Shorewall 5.1.12.2. `https://shorewall.org/`. Accessed: 2021-10-18.

[SHSK20] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. APT datasets and attack modeling for automated detection methods: A review. *Computers & Security*, page 19, 2020.

[SK13] Georgios Spathoulas and Sokratis Katsikas. Enhancing IDS performance through comprehensive alert post-processing. *Computers & Security*, 37:176–196, 2013.

[SLG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the International Conference on Information Systems Security and Privacy*, pages 108–116, 2018.

[SLW+20] Florian Skopik, Max Landauer, Markus Wurzenberger, Gernot Vormayr, Jelena Milosevic, Joachim Fabini, Wolfgang Prüggler, Oskar Kruschitz, Benjamin Widmann, Kevin Truckenthanner, et al. SynERGY: Cross-correlation of operational and contextual data to timely detect and mitigate attacks to cyber-physical systems. *Journal of Information Security and Applications*, 54:102544, 2020.

[SLW21] Florian Skopik, Max Landauer, and Markus Wurzenberger. Online log data analysis with efficient machine learning: A review. *IEEE Security & Privacy*, (01):2–12, 2021.

[SM+07] Karen Scarfone, Peter Mell, et al. Guide to intrusion detection and prevention systems (idps). *NIST Special Publication*, 800(2007):94, 2007.

[SMFDV13] Saeed Salah, Gabriel Maciá-Fernández, and Jesús Díaz-Verdejo. A model-based survey of alert correlation techniques. *Computer Networks*, 57(5):1289–1317, 2013.

[Smt] SMTP-user-enum Tool. `https://tools.kali.org/information-gathering/smtp-user-enum`. Accessed: 2021-10-18.

[SSFF14] Florian Skopik, Giuseppe Settanni, Roman Fiedler, and Ivo Friedberg. Semi-synthetic data set generation for security software evaluation. In *Proceedings of the 12th Annual International Conference on Privacy, Security and Trust*, pages 156–163. IEEE, 2014.

[ST08] Felix Salfner and Steffen Tschirpke. Error log processing for accurate failure prediction. In *Proceedings of the 1st Workshop on the Analysis of System Logs (WASL)*. USENIX, 2008.

[ST12] Sherif Saad and Issa Traore. Heterogeneous multi-sensor IDS alerts aggregation using semantic analysis. *Journal of Information Assurance and Security*, 7(2):79–88, 2012.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Ste04] John Stearley. Towards informatic analysis of syslogs. In *Proceedings of the International Conference on Cluster Computing*, pages 309–318. IEEE, 2004.

[SW99] Paul D Scott and Elwood Wilkins. Evaluating data mining procedures: techniques for generating artificial data sets. *Information and Software Technology*, 41(9):579–587, 1999.

[SWL20] Florian Skopik, Markus Wurzenberger, and Max Landauer. Decept: Detecting cyber-physical attacks using machine learning on log data. *ERCIM News*, 2020(123), 2020.

[SWL21a] Florian Skopik, Markus Wurzenberger, and Max Landauer. The seven golden principles of effective anomaly-based intrusion detection. *IEEE Security & Privacy*, 19(05):36–45, 2021.

[SWL21b] Florian Skopik, Markus Wurzenberger, and Max Landauer. *Smart Log Data Analytics: Techniques for Advanced Security Analysis*. Springer, 2021.

[Sym19] Symantec Internet Security Threat Report. Technical report, February 2019.

[TBG+11] Narate Taerat, Jim Brandt, Ann Gentile, Matthew Wong, and Chokchai Leangsuksun. Baler: deterministic, lossless log message clustering tool. *Computer Science-Research and Development*, 26(3-4):11, 2011.

[Ter] Terraform Infrastructure as Code Software. `https://www.terraform.io/`. Accessed: 2021-10-18.

[TH19] Ayman Taha and Ali S Hadi. Anomaly detection methods for categorical data: A review. *ACM Computing Surveys*, 52(2):1–35, 2019.

[TKH+17] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *arXiv preprint arXiv:1710.00811*, 2017.

[TL10] Liang Tang and Tao Li. LogTree: A framework for generating system events from raw textual logs. In *Proceedings of the 10th International Conference on Data Mining (ICDM)*, pages 491–500. IEEE, 2010.

[TLP11] Liang Tang, Tao Li, and Chang-Shing Perng. LogSig: Generating system events from raw textual logs. In *Proceedings of the 20th International Conference on Information and Knowledge Management*, pages 785–794. ACM, 2011.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[TMP17] Stefan Thaler, Vlado Menkonvski, and Milan Petkovic. Towards a neural language model for signature extraction from forensic logs. In *Proceedings of the 5th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–6. IEEE, 2017.

[TP19] Daniel Tovarňák and Tomáš Pitner. Normalization of unstructured log data into streams of structured event objects. In *Proceedings of the Symposium on Integrated Network and Service Management (IM)*, pages 671–676. IEEE, 2019.

[TR18] Wiem Tounsi and Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, 72:212 – 233, 2018.

[TSB08] Ciza Thomas, Vishwas Sharma, and N Balakrishnan. Usefulness of DARPA dataset for intrusion detection system evaluation. In *Proceedings of the Defense and Security Symposium*, volume 6973. SPIE, 2008.

[Ubu] Ubuntu 20.04. `https://ubuntu.com/`. Accessed: 2021-10-18.

[UHH+21] Rafael Uetz, Christian Hemminghaus, Louis Hackländer, Philipp Schlipper, and Martin Henze. Reproducible and adaptable log data generation for sound cybersecurity experiments. In *Proceedings of the Annual Computer Security Applications Conference*, pages 690–705, 2021.

[Vaa03] Risto Vaarandi. A data clustering algorithm for mining patterns from event logs. In *Proceedings of the 3rd Workshop on IP Operations & Management (IPOM)*, pages 119–126. IEEE, 2003.

[Vaa04] Risto Vaarandi. A breadth-first algorithm for mining frequent patterns from event logs. In *Intelligence in Communication Systems*, pages 293–308. Springer, 2004.

[VdAWM04] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[VP10] Risto Vaarandi and Kārlis Podiņš. Network IDS alert classification with frequent itemset mining and data clustering. In *Proceedings of the International Conference on Network and Service Management*, pages 451–456. IEEE, 2010.

[VP15] Risto Vaarandi and Mauno Pihelgas. LogCluster - a data clustering and pattern mining algorithm for event logs. In *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, 2015.

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.

[VPD04] Athena Vakali, Jaroslav Pokornỳ, and Theodore Dalamagas. An overview of web data clustering practices. In *Proceedings of the International Conference on Extending Database Technology*, pages 597–606. Springer, 2004.

[VS01] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Workshop on Recent Advances in Intrusion Detection*, pages 54–68. Springer, 2001.

[VVKK04] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A Kemmerer. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, 2004.

[WAAG16] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Predicting domain generation algorithms with long short-term memory networks. *arXiv preprint arXiv:1611.00791*, 2016.

[Waz] Wazuh open source security platform. `https://wazuh.com/`. Accessed: 2021-10-18.

[WC16] Chih-Hung Wang and Ye-Chen Chiou. Alert correlation system with automatic extraction of attack strategies by using dynamic feature weights. *International Journal of Computer and Communication Engineering*, 5(1):1, 2016.

[Wei09] Xu Wei. HDFS dataset . `http://people.iiis.tsinghua.edu.cn/~weixu/sospdata.html`, 2009. Accessed: 2021-10-18.

[WHL+20] Markus Wurzenberger, Georg Höld, Max Landauer, Florian Skopik, and Wolfgang Kastner. Creating character-based templates for log data to enable security event classification. In *Proceedings of the 15th Asia Conference on Computer and Communications Security*, pages 141–152. ACM, 2020.

[WLKH18] Pin-Han Wang, I-En Liao, Kuo-Fong Kao, and Jyun-Yao Huang. An intrusion detection method based on log sequence clustering of honeypot for Modbus TCP protocol. In *Proceedings of the International Conference on Applied System Invention (ICASI)*, pages 255–258. IEEE, 2018.

[WLSK19] Markus Wurzenberger, Max Landauer, Florian Skopik, and Wolfgang Kastner. AECID-PG: A tree-based log parser generator to enable log analysis. In *Proceedings of the Symposium on Integrated Network and Service Management*, pages 7–12. IEEE, 2019.

[Wor] Wordpress 5.8.2. `https://wordpress.com/`. Accessed: 2021-10-18.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[WPS] WordPress Security Scanner. `https://wpscan.com/wordpress-security-scanner`. Accessed: 2021-10-18.

[WSFK17] Markus Wurzenberger, Florian Skopik, Roman Fiedler, and Wolfgang Kastner. Applying high-performance bioinformatics tools for outlier detection in log data. In *Proceedings of the 3rd International Conference on Cybernetics (CYBCONF)*, pages 1–10. IEEE, 2017.

[WSL+17] Markus Wurzenberger, Florian Skopik, Max Landauer, Philipp Greitbauer, Roman Fiedler, and Wolfgang Kastner. Incremental clustering for semi-supervised anomaly detection applied on log data. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–6. ACM, 2017.

[WSSS16] Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Wolfgang Scherrer. Complex log file synthesis for rapid sandbox-benchmarking of security- and computer network analysis tools. *Information Systems*, 60:13–33, 2016.

[XHF+09] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the 22nd Symposium on Operating Systems Principles*, pages 117–132. ACM, 2009.

[ZD18] Ziyun Zhu and Tudor Dumitras. ChainSmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In *Proceedings of the European Symposium on Security and Privacy*, pages 458–472. IEEE, 2018.

[Zhe14] Jian Zhen. Sequence website. `http://sequencer.io/`, 2014. Accessed: 2021-10-18.

[ZHF+15] Jieming Zhu, Pinjia He, Qiang Fu, Hongyu Zhang, Michael R Lyu, and Dongmei Zhang. Learning to log: Helping developers make informed logging decisions. In *Proceedings of the 37th International Conference on Software Engineering*, volume 1, pages 415–425. IEEE, 2015.

[ZHL+19] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. Tools and benchmarks for automated log parsing. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, pages 121–130. IEEE Press, 2019.

[ZMP+13] Farhana Zulkernine, Patrick Martin, Wendy Powley, Sima Soltani, Serge Mankovskii, and Mark Addleman. CAPRI: A tool for mining complex line patterns in large log data. In *Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining*, pages 47–54. ACM, 2013.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[ZQJ16] De-Qing Zou, Hao Qin, and Hai Jin. UiLog: Improving log-based fault diagnosis by log analysis. *Journal of Computer Science and Technology*, 31(5):1038–1052, 2016.

[ZX16] Yining Zhao and Haili Xiao. Extracting log patterns from system logs in large. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshops*, pages 1645–1652. IEEE, 2016.

[ZXH11] Qiu Hua Zheng, Yi Guang Xuan, and Wei Hua Hu. An IDS alert aggregation method based on clustering. *Advanced Materials Research*, 219:156–159, 2011.

[ZXL+19] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 27th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817. ACM, 2019.

[ZZH17] Maosheng Zhang, Ying Zhao, and Zengmingyu He. GenLog: Accurate log template discovery for stripped X86 binaries. In *Proceedings of the 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 337–346. IEEE, 2017.

Die approbierte gedruckte Originalversion dieser Dissertation ist an der TU Wien Bibliothek verfügbar.
The approved original version of this doctoral thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub