



ADAPT it! Automating APT Campaign and Group Attribution by Leveraging and Linking Heterogeneous Files

Aakanksha Saha
TU Wien
Vienna, Austria
aakanksha@seclab.wien

Jorge Blasco
Univ. Politécnica de Madrid
Madrid, Spain
jorge.blasco.alis@upm.es

Lorenzo Cavallaro
University College London
London, UK
l.cavallaro@ucl.ac.uk

Martina Lindorfer
TU Wien
Vienna, Austria
martina@seclab.wien

Abstract

Recent years have witnessed a surge in the growth of Advanced Persistent Threats (APTs), with significant challenges to the security landscape, affecting industry, governance, and democracy. The ever-growing number of actors and the complexity of their campaigns have made it difficult for defenders to track and attribute these malicious activities effectively. Traditionally, researchers relied on threat intelligence to track APTs. However, this often led to fragmented information, delays in connecting campaigns with specific threat groups, and misattribution.

In response to these challenges, we introduce ADAPT, a machine learning-based approach for automatically attributing APTs at two levels: (1) the threat campaign level, to identify samples with similar objectives and (2) the threat group level, to identify samples operated by the same entity. ADAPT supports a variety of heterogeneous file types targeting different platforms, including executables and documents, and uses linking features to find connections between them. We evaluate ADAPT on a reference dataset from MITRE as well as a comprehensive, label-standardized dataset of 6,134 APT samples belonging to 92 threat groups. Using real-world case studies, we demonstrate that ADAPT effectively identifies clusters representing threat campaigns and associates them with their respective groups.

CCS Concepts

• Security and privacy → Malware and its mitigation.

Keywords

malware, advanced persistent threats, attribution, clustering

ACM Reference Format:

Aakanksha Saha, Jorge Blasco, Lorenzo Cavallaro, and Martina Lindorfer. 2024. ADAPT it! Automating APT Campaign and Group Attribution by Leveraging and Linking Heterogeneous Files. In *The 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2024)*, September 30-October 02, 2024, Padua, Italy. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3678890.3678909>

1 Introduction

In contrast to conventional malware threats, an Advanced Persistent Threat (APT) represents an adversary that pursues its objectives

over an extended timeframe, adapts to defenders' countermeasures and remains committed to maintaining the necessary level of engagement to accomplish specific goals [76]. These goals often involve exfiltrating sensitive data of economic and political significance while maintaining prolonged access within the target organization. The complexity of these attacks necessitates adversaries who are well-funded professionals and, in many cases, even have state sponsorship, enabling them to operate with the support of military or state intelligence agencies [20].

For example, in January 2021, Google's Threat Analysis Group (TAG) detected an APT campaign aiming to compromise security researchers through 0-day exploits and social engineering tactics, which involved the use of fake Twitter and LinkedIn profiles [111]. This campaign entailed months-long conversations to establish trust with the targets. Strategically luring researchers, the threat actors shared a link via Twitter, [blog.br0vvnn\[.\]io](http://blog.br0vvnn[.]io). Visiting the link installed a malicious service on the researchers' system, providing a backdoor to an actor-owned command and control server. In September 2023, TAG uncovered a new campaign, likely orchestrated by the same threat actors [50]. Both the 2021 and 2023 campaigns were coordinated operations targeting specific entities and employing similar tactics. TAG identified these threat actors as likely connected to a government-backed entity in North Korea.

As highlighted by this and similar incidents [85, 107], APT groups are well-organized entities, planning and executing multiple campaigns over time. To defend against these adversaries, the security community relies on prior knowledge of APT campaigns and groups. However, as threat campaigns and threat groups are getting more complex and sophisticated, it becomes challenging for researchers to accurately track and attribute attacks. This challenge is further exacerbated by the use of different nomenclatures and methodologies employed by security vendors to organize APT activities [91, 97]. The absence of structured, comprehensive, and easily accessible data on threat campaigns and groups hinders timely campaign identification and group attribution, thus impacting defenses [97]. For example, MITRE ATT&CK serves as a valuable resource to track threat groups and their tactics, techniques, and procedures (TTPs) [73]. However, its exclusive focus on APT groups leads to a lack of precise information on campaigns and their associated samples. Even though MITRE introduced the APT Campaign Framework in September 2022, it currently only lists 30 campaigns [72].

In response to these challenges, we introduce ADAPT ("Attributing Diverse APT Samples"), an automated machine-learning-based approach that allows for both APT *group* and *campaign* attribution. Unlike malware detection, which identifies and recognizes malicious software within a system or network, attribution seeks to



This work is licensed under a Creative Commons Attribution 4.0 International License. RAID 2024, September 30-October 02, 2024, Padua, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0959-3/24/09
<https://doi.org/10.1145/3678890.3678909>

determine if a malicious sample is associated with an APT campaign or group, providing insight into the adversary’s tactics and identity. To the best of our knowledge, ADAPT is the first attribution approach that uniquely focuses on two crucial levels of analysis, i.e., the APT campaign level and the APT group level. Attribution at the campaign level reveals specific campaign details, including tactics, techniques, and objectives, while attribution at the group level identifies the responsible entity behind the attack. An automated attribution approach that promptly clusters samples to an APT campaign or group, facilitates forensic investigations. This, in turn, enables the security community to take proactive measures, such as compiling comprehensive threat reports for Open Source Cyber Threat Intelligence (OSCTI) and alerting organizations, ultimately reducing the risk of subsequent intrusions.

Previous research on malware clustering and classification has primarily focused on identifying malware families and variants without dealing with the specific domain of APT malware [2, 11, 81, 87]. Meanwhile, research on identifying APT attacks primarily focused on network-based event analysis [7, 31, 92], which can have limited visibility on APT actions. Provenance-based anomaly detection offers a complementary approach [34, 42, 51], but suffers from dependency explosion problems and challenges in recovering complex causality relationships [37]. A handful of existing research attempted to attribute malicious samples to APT groups by relying on executable features such as basic string and code features [110], function encodings from decompiled code [69], and API call sequences [33]. Nevertheless, a notable gap remains in exploring the distinctive characteristics of APTs: Threat groups use multi-stage attack campaigns frequently leveraging a variety of file types and cross-platform samples, including 0-day exploits and custom-developed malware [17, 23, 58]. To extend the state of the art, which predominantly focused on Windows-based executables, ADAPT performs feature extraction and clustering of heterogeneous file types, including executables (e.g., PE and ELF binaries) and documents (e.g., Word documents, PDFs, and RTFs). Most notably, ADAPT uses a novel set of linking features to find connections between these different file types.

In summary, we make the following contributions:

- We compile a first-of-its-kind APT dataset of 6,134 samples encompassing heterogeneous file types from the past 17 years. We manually (re-)label and identify 92 unique APT groups in the dataset. We further create a reference dataset for APT campaigns consisting of 230 samples from 22 campaigns and 17 groups.
- We develop a novel methodology and make the first attempt at APT campaign attribution for both executable and document file types, achieving a clustering precision of 93% and 95%, respectively, on the reference dataset.
- We identify and extract a set of linking features that can facilitate sample correlation, regardless of the file types, and show promising results in APT group attribution using illustrative case studies.

Artifacts: We provide our source code, features, and labeled dataset of diverse APT samples, including executables and documents, at <https://github.com/SecPriv/adapt>.

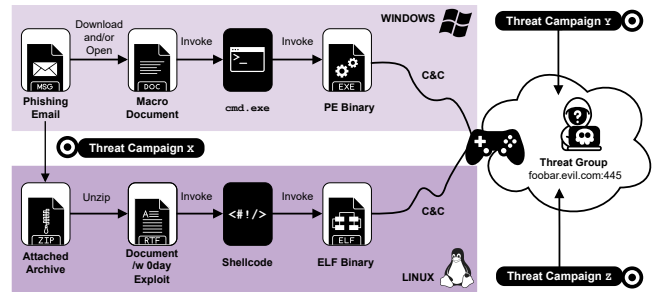


Figure 1: Simplified example based on the Sidewinder APT. A threat group conducts a number of threat campaigns that are targeting multiple platforms (e.g., Windows and Linux) using different file types, including macro-enabled or exploit-laden documents that drop platform-specific executables (PE and ELF).

2 Background and Motivation

A threat group, synonymous with the terms “threat actor” or “adversary,” refers to individuals and groups that pose threats but do not necessarily imply authorship [77]. Threat groups conduct threat campaigns, which represent specific instances of coordinated threat events associated with one or multiple threat sources, typically organized sequentially over time [78]. Each campaign usually involves a distinct set of targets, tools, and methods employed by a threat group to accomplish specific objectives.

Equipped with these definitions, we show a simplified example based on the Sidewinder APT in Figure 1. During the initial phase of the threat campaign, targeting Windows, as seen in the first highlighted block, the threat sources encompass a phishing email, a macro-enabled document, and a PE binary. The second highlighted block depicts a similar threat campaign conducted with altered attack vectors, now targeting Linux. In this case, the threat sources comprise a phishing email, a vulnerable RTF reader, and an ELF binary. The threat group is the entity that orchestrates these multiple threat campaigns targeting various organizations.

An analyst tasked with investigating the incidents similar to Figure 1 would begin by analyzing documents and executables separately, taking into consideration the intrinsic differences in layout, content, and the compilation process of executable threat sources (PE, ELF) and document threat sources. They then would attempt to correlate the following:

- (1) The macro-enabled document and the exploit-laden RTF in the document domain.
- (2) The PE binary dropped by the macro-enabled document, and the ELF binary dropped by the RTF in the executable domain.
- (3) Using the command & control (C&C) infrastructure (IP addresses, domains, and other patterns), they attempt to attribute the threat campaign X (and further campaigns Y and Z) to the threat group operating the domain *foobar.evil.com*.

In the next section, we go into more details of the Sidewinder APT as a case study to illustrate the challenges specific to correlating threat sources from multiple campaigns and associating the campaigns with their respective threat groups.

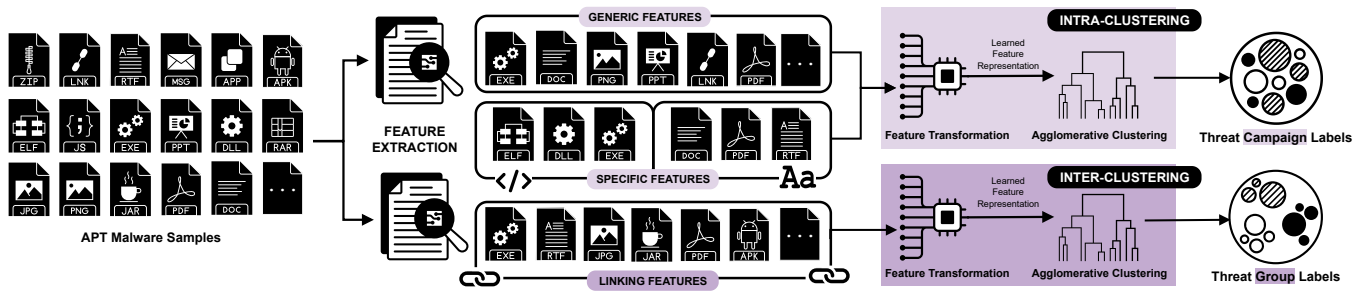


Figure 2: Overview of ADAPT. The first workflow (*Intra-Clustering*) groups executables and documents using specific and generic features to identify threat campaigns. The second workflow (*Inter-Clustering*) uses linking features across all file types to identify threat groups.

2.1 Motivational Case Study: Sidewinder

Sidewinder, also known as Rattlesnake and T-APT-04 [46, 70], is a threat group believed to be based out of India. In particular, we focus on how the group uses heterogeneous files and platforms, adapts their campaigns based on their goals, and the issues around shared tooling and code reuse between groups.

Heterogeneous Files and Platforms. Sidewinder uses malicious documents as one of the most common infection vectors. In addition, analysts have also observed the use of various file types in different campaigns for delivering malicious payloads [23, 46]. These file types include LNK files used to download RTF files that subsequently drop JavaScript files or ZIP files containing LNK files that download HTA files with JavaScript. Further, these files vary over the years and often evolve with each campaign in an attempt to complicate analysis and evade detection. Moreover, some of the document file types are cross-platform and embed platform-specific exploits. A noteworthy similarity lies in the group’s utilization of the Microsoft Office memory corruption vulnerability CVE-2017-11882 [75] as a means to initiate the compromise on target hosts. Additionally, Sidewinder has also been spotted using the Binder exploit to attack mobile devices [106]. This demonstrates how the group strategically employs multiple file types and exploits multiple platforms, utilizing 0-day vulnerabilities, to significantly increase the likelihood of a successful attack. While the utilization of a variety of file formats, besides the widespread Windows PE binaries, is a technique employed by both APTs and conventional malware threats, as noted in previous studies [13, 93, 99], our research is specifically focused on APTs and their distinctive characteristics.

Varied and Persistent Campaigns. Sidewinder uses spear phishing to obtain credentials from targeted organizations [103] including the distribution of maliciously crafted documents containing executable payloads [23]. The themes and topics of phishing pages and malicious documents adapt to the campaign’s objectives. These objectives can involve gaining sensitive information related to COVID-19 research or territorial disputes involving Nepal, Pakistan, India, and China [46]. These variations have been described as the group’s effort to craft unique campaigns based on the target organization and global affairs [106].

In 2022, Sidewinder introduced a new custom tool (SideWinder .AntiBot.Script) to redirect victims to download the initial payload from a compromised website [36]. Sidewinder’s persistence and varying tactics across multiple targets complicate attribution.

Security vendors tracking the same APT group from different campaign perspectives often generate overlapping or fragmented information [86, 103], leading to an incomplete understanding.

Shared Similarities. The utilization of common components in the attack chain, code reuse, and the sharing of toolkits among different threat groups also create challenges for researchers in accurately attributing a sample to its respective threat group [10]. For instance, in at least two investigations, code sharing among APT groups with opposing targets was highlighted [18, 101]. Initially, analysts associated a set of samples with either Sidewinder or Donot groups [24]. However, on further investigation, these samples were conclusively linked to a third APT group, Transparent Tribe [71], also known as APT36. Transparent Tribe has been active in South Asia for over five years, primarily targeting the Indian government and military organizations. Despite different target regions (India for Transparent Tribe and Pakistan for Sidewinder and Donot), these groups reused portions of the Visual Basic Analysis (VBA) code. The code similarity among the groups resulted in inconsistent and erroneous attribution. To further complicate the matter, many APT groups consist of subgroups, each assigned specific tasks. The collaboration among various APTs makes it challenging to confidently track and attribute actions to a single entity [9].

3 Our Approach: ADAPT

Motivated by the challenges detailed above, we propose ADAPT, an attribution approach specifically designed to handle heterogeneous file types. ADAPT streamlines the attribution process by clustering samples at two levels. First, at the *campaign level*, it attributes samples based on their specific tactics and techniques. This means that regardless of the campaign’s evolving nature or changing names over time, ADAPT’s focus on the inherent functionalities enables us to associate specific samples with an APT campaign. Second, at the *group level*, ADAPT considers infrastructure and operational traits, providing insights into the broader context within which campaigns operate and attributes samples based on the characteristics of the APT group. Clustering malware samples is a routine task performed by malware analysts, as demonstrated by Yong et al. [113]. This highlights the importance of automating forensic investigations of APTs through campaign- and group-level clustering

Figure 2 illustrates our approach for analysts using ADAPT during security incident investigations. Instead of manual correlation

of diverse threat sources, analysts can leverage ADAPT to *automatically* identify if recovered malicious artifacts exhibit indicators of known or ongoing APT campaigns stored in the APT dataset (see Section 4). ADAPT’s feature extraction first identifies the file type and extracts relevant static attributes. Subsequently, for executables and documents, ADAPT’s *Intra-Clustering* attributes the samples to campaigns (see Section 5). This categorization streamlines classification and provides insights into campaign tactics and techniques, facilitating sample analysis. ADAPT’s *Inter-Clustering* enables grouping samples by the threat actor (see Section 6) leveraging linking attributes to facilitate correlation based on distinct characteristics and operational patterns.

Note that we maintain a dedicated clustering step for group and campaign attribution for two primary reasons. First, as highlighted in our motivating case study, threat groups execute a wide range of campaigns targeting various organizations. These campaigns, initially unattributed, become linked to an APT group following thorough analysis and information gathered from prior campaigns [63, 108]. Second, APT attacks typically involve multiple file types as shown in our motivational study. Therefore, having a dedicated group attribution process that accommodates all artifacts associated with different campaigns proves valuable for linking and associating samples. This, in turn, simplifies attributing and prosecuting recurring APT attacks, as exemplified in the FBI indictments of APT10, APT29, and APT41 [29, 30, 38].

4 APT Dataset

In this section, we describe our APT sample collection process, as well as the labeling process that we incorporated to ensure consistency within the dataset.

4.1 Group-labeled Dataset

4.1.1 Data Collection. To collect APT sample hashes, we use AlienVault’s DirectConnect API [4], leveraging their comprehensive threat intelligence database. By specifically querying for APT-related pulses, we retrieved 5,990 unique SHA256 hashes associated with a minimum of 172 threat groups. The threat group labels are crowd-sourced from the AlienVault community and we collect them along with the APT hashes. To further expand our dataset and ensure it includes the latest hashes we extract unique SHA256 hashes from threat reports published by Unit42 [107] and Mandiant [15] between January 2022 and March 2023. We chose Unit42 and Mandiant as they are reputable sources known for their reliable and up-to-date threat reports. Through this effort, we discovered 465 hashes attributed to ten threat groups. Notably, most of these hashes were explicitly linked with the Gamaredon APT group, highlighting their prominent role in the Russia-Ukraine conflict.

Since AlienVault only provides hashes and Indicators of Compromise (IoCs), we use VirusTotal [109] to download the corresponding 6,455 samples. We also query VirusTotal to obtain analysis reports for the downloaded samples and extract metadata such as the file type, creation date, and first submission date. The most recent sample in our dataset was first submitted in March 2023, while the earliest sample submission date was in May 2006. Note that the available number of samples for APTs is significantly smaller compared to generic malware due to their targeted nature. Unlike

commodity malware with its indiscriminate victim selection, APT malware focuses on high-value targets, aiming for significant impact (often espionage or sabotage) and avoids mass distribution. This limited presence makes them scarce for collection and analysis, as noted in prior studies [32, 89].

4.1.2 Data (Re-)Labeling. 2,260 samples (35.01%) in our dataset have more than one APT name or alias, and 239 (3.7%) samples are labeled with five or more group names, with the highest number of labels being 15 for two samples. For instance, some samples are tagged as ‘Bluenoroff,’ ‘AppleJeus,’ or ‘Hidden Cobra’ which are all aliases originating from different campaigns of Lazarus. This highlights the challenges analysts face when correlating campaign and group data. To address the lack of standardized labels, we conduct a thorough revision and relabeling process using Malpedia [57] and MITRE [73] and establish consistent group labels. In instances when a reliable label could not be confirmed due to disagreements between Malpedia and MITRE, two researchers independently reviewed threat reports from Unit42 and Mandiant. They each assigned potential threat group labels and then discussed and resolved any mismatches or conflicts. This process included addressing the following issues:

Consistency of Existing Labels: Aliases. Numerous samples are tagged with aliases representing the same APT group, e.g., ‘Refined Kitten’ and ‘Elfin,’ which correspond to APT33. We standardize these aliases following classifications by Malpedia and MITRE.

Consistency of Existing Labels: Umbrella Names. We eliminate text-based variations and adopt a consistent naming convention of APTXX, FINXX, or TAXXX for the groups whenever feasible. For instance, we assign the name APT32 for ‘OceanLotus’ and ‘Sea Lotus,’ and APT43 for ‘Kimsuky,’ while APT24 encompasses ‘Pitty Putter’ and ‘Pitty Tiger,’ following Malpedia’s classification.

Consistency of Existing Labels: Non-unique Names. We manually review non-unique names like ‘Transparent Tribe,’ ‘transparenttribe,’ or ‘Transparent Tribe Group,’ and assign the name ‘TransparentTribe’ as the uniform representation.

Outliers: Non-APT Samples. We came across 122 (1.89%) samples linked to the FireEye Red Team tools that were stolen during the SolarWinds attack [105]. We classify them as ‘NotAPT’ since they are not actual attacker tools, but instead part of the data exfiltrated by the attackers. We also came across ransomware families originally labeled as ‘Egregor’ or ‘Maze Team’ and classified them as ‘NotAPT.’ Our decision is based on the observation that these ransomware families are employed by multiple cybercrime groups for financial gain and are not typically associated with a specific APT group. We remove these samples from our dataset.

Unlabeled Threat Groups. We identified 44 (0.68%) samples for which we could not determine the APT group label with certainty due to two main factors. First, certain threat reports utilize internal operation names that are not associated with any specific groups by Malpedia or MITRE. Second, some threat reports discuss the likely origin of the attack without providing a conclusive group name. We retain these samples without labels in our dataset.

4.1.3 Dataset Characteristics. After following the approach outlined above and excluding 321 (4.97%) ‘NotAPT’ samples, we are left with a dataset of 6,134 samples. The (re-)labeling effort identified 92

distinct APT groups, resulting in a decrease of 80 tags compared to the initial 172 group tags extracted from AlienVault. Table 1 shows the sample count for the top 15 APT groups in our dataset, along with the number of aliases provided by Malpedia (lower bound). The median sample count for APT groups in our dataset is 24.

Diversity of File Types. To extract file type information, we rely on the VirusTotal file analysis reports associated with the samples. Through experimentation, we observed that VirusTotal consistently provided the most accurate and standardized file type information compared to other methods, such as libmagic [82]. Table 2 provides an overview of the number of distinct file types in our dataset, as well as whether ADAPT treats them as executables or documents (see Section 5). Our dataset contains a total of 3,603 (58.73%) executable binaries, including samples targeting Windows, Linux, and macOS, the latter of which have received limited attention from the research community so far [21, 54]. The document file class is another understudied domain [93] and consists of 1,611 (26.26%) files, includes formats such as Microsoft Word, Excel, PowerPoint, RTF, PDF, and ZIP, Note that we consider ZIP files as documents because our feature extraction is capable of extracting attributes from ZIP files that contain valid document formats. Finally, our APT dataset also includes Android apps (APK), Windows shortcuts (LNK), and script files, as well as 152 unknown file formats.

4.2 Campaign-labeled Dataset

Compared to datasets labeled by threat groups, the scarcity of publicly available, structured, and comprehensive information for threat campaigns presents an even bigger challenge. To address this gap and build a reference dataset for threat campaigns, we use MITRE’s APT Campaign Framework [72]. This campaign tracking data lists campaigns with standardized identifiers (CXXXX format). For example, the C00024 campaign is associated with the SolarWinds compromise by the threat group APT29.



We extracted MD5, SHA1, and SHA256 file hashes from MITRE reports and downloaded the corresponding samples using VirusTotal. However, since not all samples were available on VirusTotal, we expanded the dataset with samples mentioned in threat reports from Mandiant [59], which offered in-depth coverage of APT campaigns as recent as March 2024. This combined effort resulted in the successful download of 231 samples. To ensure accuracy, we thoroughly examine the corresponding threat reports for each sample, identifying and removing any false positives and assigning the appropriate campaign label and group label. Our final dataset consists of 230 samples representing malware from 22 campaigns attributed to 17 distinct groups. These samples encompass various file types, including 142 executable binaries (139 PE, 2 MachO, 1 ELF) and 62 documents (34 DOCX, 9 ZIP, 8 DOC, 5 PDF, 4 HWP, 1 XLSX). We make both this *campaign-labeled* as well as the above *group-labeled* dataset publicly available.














5 APT Campaign Attribution

Unlike detection tasks that solely focus on identifying maliciousness, our research automates the analyst’s workflow for malware campaign classification. This process involves grouping attacks based on shared characteristics, necessitating information beyond simple “benign” vs. “malicious” attributes. Establishing connections

Table 1: Top 15 threat groups in our APT dataset.

Threat Group Label	Number of Aliases	Sample Count
Lazarus	29	527 (8.59%)
Gamaredon	11	446 (7.27%)
TransparentTribe	9	403 (6.56%)
WizardSpider	3	401 (6.53%)
TA505	9	307 (5.00%)
FIN7	8	293 (4.77%)
APT41	16	278 (4.53%)
APT1	11	251 (4.09%)
APT29	15	224 (3.65%)
Turla	21	203 (3.30%)
Kimsuky	5	173 (2.82%)
APT28	23	169 (2.75%)
APT32	15	147 (2.39%)
Sidewinder	4	133 (2.16%)
APT34	10	126 (2.05%)
Others	-	2,054 (33.48%)
Total	-	6,134

Table 2: Different file types in our curated APT dataset. The file class indicates whether ADAPT specifically handles files as executables () , documents () , or only extracts generic features.

File Type	File Class	Platform	Total #
Windows Portable Executable (PE; EXE)		Windows	2,516 (41.01%)
Windows Portable Executable (PE; DLL)		Windows	1,019 (16.61%)
OOXML Document		Cross-Platform	286 (4.66%)
Microsoft Word Document (DOC[X])		Cross-Platform	262 (4.27%)
Rich Text Format (RTF)		Cross-Platform	245 (3.99%)
Microsoft Excel Spreadsheet (XLS[X])		Cross-Platform	239 (3.89%)
Android Application Package (APK)		Android	227 (3.70%)
Windows shortcut (LNK)		Windows	223 (3.63%)
ZIP Archive		Cross-Platform	129 (2.10%)
OOXML Spreadsheet		Cross-Platform	104 (1.69%)
Text		Cross-Platform	91 (1.48%)
JavaScript (JS)		Cross-Platform	66 (1.07%)
Hypertext Markup Language (HTML)		Cross-Platform	65 (1.05%)
Portable Document Format (PDF)		Cross-Platform	60 (0.97%)
Visual Basic for Applications (VBA)		Cross-Platform	56 (0.91%)
Powershell		Windows	44 (0.71%)
Executable and Linkable Format (ELF)		Linux	39 (0.63%)
Adobe Flash		Cross-Platform	38 (0.61%)
RAR Archive		Cross-Platform	38 (0.61%)
Mach Object (MachO)		macOS	29 (0.47%)
Hangul Word Processor (HWP)		Cross-Platform	27 (0.44%)
Microsoft PowerPoint (PPT[X])		Cross-Platform	15 (0.24%)
Others	-		355 (5.78%)
22+ File Classes		4+ Platforms	6,134

with known threat campaigns (related executables and documents) serves as a crucial starting point for prioritizing investigation in identifying attacker objectives and potential consequences [65, 113].

As shown in Table 2, executables and documents constitute the majority of our dataset. We extract features specifically tailored for these prevalent file types. Beyond this *file-specific features*, we enrich both executable and document representations with *generic features* derived from their string content. These generic features, extractable without specialized parsing, target the detection of malicious techniques and patterns and apply to various file types. While we do not perform specific feature extraction for APK and LNK files, as well as scripts, in this work, the incorporated generic features are applicable to these file types as well. We acknowledge the potential for future exploration in this area. Our approach

prioritizes identifying the most critical features that characterize campaign-level similarities, rather than creating an exhaustive list of indicators. Focusing on these key features enables interpretable and automated campaign attribution across diverse malicious file types, as demonstrated by our case studies (see Section 9).

Executable Specific Features (EXF). Leveraging prior research in malware detection [6], [21], [99], and classification [2], [66] we select features that provide generalizability across the executable file types while enabling fast clustering. Our decision not to use type- and platform-specific features collected through more heavy-weight static and dynamic analysis (e.g., disassembled executables, call graphs, and system-level execution traces) allows us to compile a simple yet representative feature set. This reduces limitations associated with missed run-time behaviors due to the unavailability of attacker infrastructure or dependence on specific host artifacts.

We extract a comprehensive set of features from executables, such as section names, libraries, and imported/exported functions. We use LIEF [102] to parse and extract these features from the 3,603 (3,535 PE files, 39 ELF files, and 29 MachO) executable binaries. Because imported functions produce a large number of values with little discriminative power, and libraries, and section names are common to many different types of malware, we narrow down the feature set to focus on *exported functions* and the *configuration version* for PE executables. These features demonstrated discriminative potential in our initial experiments on a subset of malicious samples from distinct campaigns operated by the same threat actor.

For PE, ELF, and MachO executables, we extract 15,047 unique exported functions from 3,603 binaries, such as `DllRegisterServer`, `DllUnregisterServer`, `runtime.gosched_m`, `FileRipper`, `net.dnsDefaultSearch`, and `runtime.prefetcht0`. Additionally, we identify 11 unique configuration values for the PE executables, such as `WIN_VERSION.SEH`, `WIN_VERSION.WIN_8_1`, and `WIN_VERSION.WIN10_0_15002`, which can provide indicators about the attacker’s build system used in a specific campaign.

Document Specific Features (DCF). From the document file types, we extract features including macros, obfuscated strings, document author (if available), application language, and suspicious keywords. Similar to prior research on document analysis [45, 68], we use the open-source Python package `oletools` [47] to parse and extract malicious content from document file formats, including ZIP. For ZIP archives, we iteratively parse and extract attributes from individual files. Consequently, we obtain features from 1,552 files (96.33%) out of a total of 1,611 document samples. Among those, we identify two informative and distinctive features for our clustering algorithms: the *Application Language Code* and the list of *Suspicious Keywords* with a total of 2,578 unique values. Suspicious keywords include malicious patterns such as auto-executable macros, VBA keywords used by malware, anti-sandboxing, and anti-virtualization techniques identified through pattern matching on macro scripts embedded in document formats (e.g., Word and Excel documents). Examples of suspicious keywords include `keywords-AMANICRYPTED.exe`, `keywords-PrivateFunctionF()`, and `keywords-AutoExecute`. Finally, examples of language codes include 1251: ANSI Cyrillic (Windows), ANSI/OEM Korean (Unified Hangul Code), which are standardized codes used to identify specific languages within the application.

Generic Features: Capabilities (CAP). We use features to detect and extract distinct characteristics and capabilities in malicious files. To do this, we leverage the targeted and effective set of YARA rules developed by the Malcat Community [56]. These rules are designed to detect modular capabilities within the program, such as code injection, remote thread routines, privilege escalation for lateral movement, persistence using scheduled tasks, as well as packers. With a comprehensive set of 108 rules, we successfully identified at least one rule for 4,026 samples (65.63%). We also considered using the Yara-Rules [112] and Elastic’s security detection rules [96] but found that they often exhibit noise or insufficient coverage.

Following the extraction, we use one-hot encoding to transform the categorical features from both executables and documents. These features collectively form the set S of all categorical features, $S = S_1, S_2, S_3, \dots, S_i$. For instance, consider the feature “Capabilities”, (S_1), with values `MSVC_2017_linker`, `MSVC_2017_rich`, and `DownloadUsingWinHttp`, where $i = 1$ and $k = 3$ unique values. To represent each value, we employ a binary vector S_{ik} where, $S_{ik} \in \{0, 1\}$. We consider the entire set of unique values, assigning a value of 1 if a particular element is present in the sample.

Generic Features: Strings (STR). Extracting strings from malware samples can give insights into a threat group’s preferred syntactic construction and vocabulary, which can be used to identify the associated campaigns. We use FLOSS [62] to extract all possible ASCII and UTF texts from 6,114 samples (99.67%). In addition to extracting embedded ASCII and UTF strings, FLOSS is also capable of identifying stack strings (strings constructed on the stack at run time) and decoded strings (strings decoded in a function) from PE files, which can enhance the basic static analysis of malicious files.

We preprocess and filter the extracted strings to remove numeric characters, special characters, non-printable characters, spaces, and stop words. We also perform Unicode normalization to decompose Unicode characters into their individual components, ensuring consistent representation (e.g., 隨 \rightarrow U+968F). After preprocessing, we employ the `CountVectorizer` technique with n -grams. This method calculates the frequency of string tokens in the document. The effectiveness of this approach has been demonstrated in prior research on malicious application attribution and malware source code identification [43, 88]. In our implementation, we set the parameters of the vectorizer to use n -grams of size 1 to 3, with a maximum of 10,000 features. After obtaining the vectorized representation of the string tokens, we normalize the values of each token. This step plays a crucial role in integrating the string features with other numerical and categorical features, ensuring that all features are scaled within the range of 0 to 1 for the clustering task.

6 APT Group Attribution

Achieving both campaign and group attribution necessitates addressing the inherent heterogeneity of artifacts employed across multiple APT campaigns by a specific group (see Section 2). Thus, our feature selection process leverages domain expertise to identify attributes that effectively link these diverse threat sources with group-level signatures. For instance, in the supply chain attack uncovered by Mandiant [61], the analysis of 11 distinct campaigns

targeting diverse sectors revealed the use of shared resources (digital certificates, infrastructure, development tools) across all campaigns, indicating a single threat group. Inspired by real-world investigations [60, 61, 107], we focus on extracting linkable features commonly used by analysts to connect malicious artifacts across campaigns to threat actors. ADAPT automates the process of extracting *linking features* beyond basic file analysis to facilitate group attribution in APT incidents.

Pattern-based (PAT). To identify the linking characteristics, we first extract specific patterns from the string features, part of our generic features (see Section 5). These patterns include IP addresses, URLs, authentication keys, API keys (e.g., Slack, Gmail, and AWS), embedded MD5, SHA1, SHA256 hashes, Bitcoin addresses, email addresses, and Unix and Windows file paths. We choose these features as they can reveal important traits about the operating threat group. For example, the presence of a specific file path like `C:\Windows\System32\drivers\ within an APT sample could indicate a threat group’s preference for installing malicious drivers in a specific location. We leverage a set of 24 regular expressions to automate pattern matching across our samples. Out of 6,134 samples in our dataset, we found one or more patterns in 4,506 samples (73.45%).`

Infrastructure (INF). After extracting URLs and IP addresses statically from the sample’s string content, we then use them to collect more detailed infrastructure information. This includes the BGP prefix, autonomous system number (ASN), country code, certificate fingerprint, and issuer organization. Leveraging the Censys search engine API [26], we query these identified IP addresses and URLs for granular host and domain details. Note that we use the ‘datetime’ query to narrow our results on those matching the first submission date found in the VirusTotal file report. This approach serves two benefits. First, it helps us focus on a limited set of results, as some domains can generate a large number of certificate and host results. Second, by targeting the period when the sample was most likely to be active, we ensure that the results are more relevant to the threat group’s activity. Moreover, we exclude the top 500 domains (e.g., `wordpress.com`, `europa.eu`, `drive.google.com`) and a list of reserved IP blocks (e.g., `0.0.0.0/8`, `127.0.0.0/8`) from our search to ensure that we focus solely on identifying domains and IPs specific to threat groups. Following this process, we were able to extract infrastructure features for 2,345 samples (38.22%) in our dataset. Although these features were not available for all samples, we included them in our clustering process as they complement the pattern-based features. A complete list of the pattern-based and infrastructure features is available as part of our artifact.

To transform the raw *linking* features, we use a sentence transformer to generate text embeddings for clustering. Specifically, we use the pre-trained sentence transformer model, trained on a large dataset of 215 million (question, answer) pairs from various sources [83]. This semantic search model encodes textual features into a compact vector space, allowing it to capture subtle variations in complex patterns. These generated embeddings are then compared, for instance, using cosine similarity, to identify sentences with similar meanings. To illustrate this process, consider four samples, each containing the URLs: `http://a0711854.xsph.ru`, `http://a0713099.xsph.ru`, `http://a0714424.xsph.ru`, and

`http://ca.mtin.es`, respectively. Employing the sentence transformer model, we generate encodings for these URLs and set a similarity threshold of 0.8. As a result, we identify the first three URLs as highly similar, leading us to assign these three samples to the same bucket, indicating their similarity based on the closely related URLs. The last sample remains in a separate bucket. We tried different pre-trained models [27, 28] and thresholds, and found that the best combination was the “multi-qa-MiniLM-L6-cos-v1” model with a 0.8 similarity score. We extend this approach to certificate issuer organizations (e.g., “Verisign,” “DigiCert Inc.”), email addresses, and Windows and Linux file paths.

Further, to eliminate overly common features, we analyzed their frequency of occurrence in the dataset. We perform feature selection on a subset of samples and determine thresholds by examining the distribution of feature frequencies in the validation set. Specifically, we identify a threshold of 0.75. To retain the most meaningful features, we calculate the percentage of samples in which each feature appears and remove those present in more than 75% of the samples. Common examples of such elements include generic cloud URLs, IP addresses, country codes, and common certificate names. This approach refines the dataset by highlighting rare and potentially more meaningful relationships between the samples.

7 Clustering Implementation

Due to the general lack of reliable ground truth labels for APT malware, accurately classifying samples is challenging, as demonstrated by our relabeling efforts (see Section 4). Therefore, unsupervised clustering becomes a feasible solution. We use agglomerative hierarchical clustering [94], a method frequently employed in malware clustering [11, 80, 81]. Hierarchical clustering’s strength lies in its ability to identify clusters of arbitrary shapes, making it well-suited for capturing complex relationships within the APT domain. The agglomerative clustering recursively merges similar clusters, resulting in a dendrogram-like structure. Initially, each data point is treated as a separate cluster (referred to as a leaf), and the algorithm computes the distance between these individual clusters.

Our experiments showed that the agglomerative approach can be computationally expensive for large feature spaces due to its recursive nature. To address this, we incorporate autoencoders to learn a latent representation of our features. Autoencoders offer the ability to learn compact representations of input data by capturing the underlying structure and removing noise or irrelevant information [98]. This compressed representation allows the clustering algorithm to converge faster. The autoencoder architecture employed in our approach consists of four layers: an input layer, two hidden layers, and an output layer. The input layer has the same number of units as the transformed feature set, while the hidden layers comprise 32 and 16 units, respectively. To activate the hidden layers, we employ the rectified linear unit (ReLU) activation function, due to its computational efficiency [39]. The output layer mirrors the input layer in terms of the number of units and employs the sigmoid activation function.

For optimal clustering results, we perform cluster validity analysis, using the metric Sum of Squared Errors (SSE) [79]. We iteratively merge clusters by selecting the below configuration that results in the lowest change in SSE:

- *Distance Metrics*: We consider the Euclidean and Manhattan distances to measure the dissimilarity between clusters.
- *Compute Full Tree*: We set the parameter ‘compute_full_tree’ to True in order to compute the complete hierarchical tree without early pruning.
- *Linkages*: We iterate over different linkage algorithms, including ward, complete, and average, to determine the optimal method for merging clusters.
- *Number of Clusters*: We explore various numbers of clusters to find the most suitable configuration.

Further, to identify the optimal number of clusters, we use the elbow method. This method involves running the clustering algorithm for different values of clusters (k) and plotting the SSE for each run. The elbow point, where the change in the SSE starts to diminish significantly, indicates the optimal k for our clustering tasks. We also experimented with other clustering algorithms such as HDBSCAN and K-means, achieving comparable results.

We employ the *ADAPT Intra-Clustering* for APT campaign attribution (see Section 5 and top of Figure 2) for our executable and document samples. This approach groups samples within dominant file domains based on their shared similarities. Subsequently, the *ADAPT Inter-Clustering* (see Section 6 and bottom of Figure 2) for APT group attribution groups samples across all files incorporating the transformed linking features (PAT & INF) extracted from string content. It is worth noting, that our approach incorporates all file types from the dataset (see Table 2). Although some files may not yield readily identifiable patterns through static features, their inclusion allows for a more holistic analysis, and the potential discovery of subtle connections. However, this inclusivity can lead to singleton clusters or misclustering. To mitigate this, we employ an analyst-defined threshold for excluding samples with insufficient features, ensuring focus on the most informative samples for accurate threat group identification.

8 Evaluation and Results

In this section, we evaluate ADAPT’s performance and effectiveness using the campaign-labeled dataset curated from MITRE information (discussed in Section 4.2). We report quantitative performance metrics for the attribution tasks in Section 8.1. We further explore the relative importance of different features for executables and documents in attributing threat campaigns in Section 8.2.

Experimental Setup. We implemented ADAPT using the Python programming language. We performed all experiments on a Windows 11 Pro machine with the following specifications: 12th Gen Intel(R) Core(TM) i9-12900KS, 3400 Mhz, 16 Core(s), 24 Logical Processor(s), and 32 GB RAM. To maintain a controlled research environment, the machine is connected to a dedicated router with no other devices connected to the network. Furthermore, to prevent interference from Windows Defender Antivirus, we place the dataset directory in an exclusion folder with real-time monitoring disabled [67]. The dataset comprises 6,134 samples with raw extracted features stored in JSON files, requiring 15.6 GB of storage.

8.1 Cluster Validity Analysis

Assessing the results of unsupervised clustering algorithms is challenging due to the lack of ground truth labels. There is no standard method for validating the output of clustering results [40]. Clustering algorithms aim to group similar objects together based on metrics such as distance between and within clusters. However, a key challenge that arises is determining the optimal number of clusters. As mentioned in Section 7, we determine the optimal number of clusters by minimizing the Sum of Squared Errors (SSE). SSE is calculated by summing the squared differences between the data points in a cluster and the cluster’s centroid. A lower SSE indicates a more compact and dense cluster, implying that the objects within the cluster are closely related [100]. SSE measures the error or deviation within clusters based on the internal structure of the data, without using external ground truth labels. However, it is also helpful to analyze the clustering results by comparing them to existing ground truth labels [11].

Precision and Recall. We use the campaign-labeled dataset derived from MITRE information (discussed in Section 4.2) to validate our clustering results. This dataset, allows us to measure the level of agreement between the clusters we obtain and the information associated with the clustered samples. Bayer et al. [11] proposed to use precision and recall by establishing a mapping between the outcomes of their system-level behavioral clustering system and the *reference* clustering. In a similar vein, Perdisci et al. [80] presented an alternative method to evaluate the credibility of clustering results, concentrating on assessing the cohesion within individual clusters and the separation between different clusters. Based on these previous studies, we propose a method to normalize labels between clustering results and the reference cluster.

Given a dataset of samples $S = \{S_1, S_2, S_3, \dots, S_n\}$, comprising n samples identified by their SHA256 hash, and a set of t distinct ground truth/reference clusters $T = \{T_1, T_2, T_3, \dots, T_t\}$, we apply a clustering algorithm to the dataset S resulting in a set of predicted clusters $C = \{C_1, C_2, C_3, \dots, C_c\}$. Here, c represents the total number of distinct predicted clusters, and each predicted cluster C_i contains an arbitrary number of samples and is assigned a cluster label i .

To normalize cluster labels, we take the dataset S , the predicted clusters C , and the ground truth clusters T as inputs. For each predicted cluster C_i within C , we retrieve the subset of samples S_i from the dataset S that belong to that specific predicted cluster C_i . Subsequently, we count the occurrences of each ground truth cluster label t within S_i and identify the label with the highest count, denoted as t_{majority} . We assign t_{majority} as the normalized cluster label N_i for the predicted cluster C_i . We repeat the normalization process for all predicted clusters in C , generating the collection of normalized cluster labels $N = N_1, N_2, N_3, \dots, N_c$. Using normalized cluster labels N and the ground truth labels T , we define:

True Positives (TP): The number of samples that are correctly assigned to the same cluster both in the normalized cluster labels N and the ground truth labels T .

$$TP = \sum_{i=1}^c \sum_{j=1}^t I(N_i = T_j) \cdot I(C_i \cap T_j \neq \emptyset)$$

False Negatives (FN): The number of samples that are incorrectly assigned to a different cluster in the normalized cluster labels N but belong to the same cluster in the ground truth labels T .

$$FN = \sum_{i=1}^c \sum_{j=1}^t I(N_i \neq T_j) \cdot I(C_i \cap T_j \neq \emptyset)$$

False Positives (FP): The number of samples that are incorrectly assigned to the same cluster in the normalized cluster labels N but belong to a different cluster in the ground truth labels T .

$$FP = \sum_{i=1}^c \sum_{j=1}^t I(N_i = T_j) \cdot I(C_i \cap T_j = \emptyset)$$

Here $I(x)$ denotes an indicator function that takes an argument x and returns 1 if x is true, and 0 otherwise.

Clustering Results. Using the above-defined metrics, we calculate precision, recall, and F1-score to evaluate the performance of our campaign and group attribution tasks (see Section 5 and 6) on the reference dataset. Further, we report the clustering metrics, Silhouette coefficient (SC) [95], and the SSE. SC, a common metric for unsupervised tasks, ranges from -1 to 1. It measures the average distance between a sample and its assigned cluster compared to the distance to the next nearest cluster. Values near 0 indicate clusters with some overlap, while negative values suggest samples being placed in the wrong cluster. Our goal is to identify the number of clusters that minimize SSE and have a positive SC.

Among the executable samples, we achieve a precision of 0.93, a recall of 0.92, and a combined F1-score of 0.91. The SSE is 1.45 and the SC is 0.50. Document samples demonstrate higher precision (0.95) and recall (0.94), resulting in an F1-score of 0.92. The SSE for documents is 0.72 with an SC of 0.36. We hypothesize that the clustering performance of documents is better than that of executables because they tend to exhibit consistent, unique patterns across campaigns. In contrast, executables often use obfuscation techniques and display polymorphic behavior, making it difficult to identify distinguishing features for clustering. However, the 93% precision for executables demonstrates that ADAPT was able to effectively distinguish between samples from different campaigns. For the threat group attribution task, we achieve a precision of 0.92, a recall of 0.89, and an F1-score of 0.89. While this F1-score reflects relatively good performance, the SC of 0.41 obtained with the lowest SSE of 2.70 highlights the challenges of accurately correlating diverse samples across executable and document domains. Further, we report the number of clusters identified in our reference clustering consisting of 230 samples with 22 campaigns and 17 threat groups. For the threat campaign attribution task involving executables, we identified 18 clusters with a precision of 93% and a recall of 92%, while for documents, we identified 9 clusters with a precision of 95% and a recall of 94%. For the group attribution task, we identified 15 clusters with a precision of 92% and a recall of 89%.

8.2 Feature Importance

In the context of unsupervised clustering algorithms, we assess the suitability of different features by analyzing their impact on precision and recall in our reference dataset. Table 3 shows the

Table 3: List of feature categories for executables and their performance on the campaign-labeled reference dataset. We evaluate combinations of *Executable Specific Features* (EXF) and *Generic Features*, including *Capabilities* (CAP) and *Strings* (STR).

Feature Category	# Features	Precision	Recall	F1-score
EXF	22,042	0.85	0.72	0.70
EXF + CAP	22,099	0.88	0.87	0.85
EXF + STR	77,697	0.91	0.90	0.89
EXF + CAP + STR	77,759	0.93	0.92	0.91

Table 4: List of feature categories for documents and their performance on the campaign-labeled reference dataset. We evaluate combinations of *Document Specific Features* (DCF) and *Generic Features*, including *Capabilities* (CAP) and *Strings* (STR).

Feature Category	# Features	Precision	Recall	F1-score
DCF	85	0.88	0.84	0.79
DCF + CAP	142	0.93	0.93	0.92
DCF + STR	44,035	0.92	0.91	0.91
DCF + CAP + STR	44,097	0.95	0.94	0.92

importance of different feature categories for clustering executable samples. The importance of a specific feature is determined by the degree to which it increases the overall F1-score. From the table, we can see that by solely incorporating the executable-related features (EXF) like exported functions, we observe a relatively low F1-score of 0.70. However, combining these with the detected capabilities (EXF + CAP) and string features (EXF + STR) significantly improves the F1-score to 0.85 and 0.89 respectively.

Similarly, Table 4 shows the importance of different features for accurate clustering of document samples. Notably, relying solely on document-related features (DCF), like suspicious keywords, yields a relatively low F1-score of 0.79. However, incrementally incorporating modular capabilities (CAP) and strings (STR) into the feature set results in an F1-score improvement, closely approaching the highest F1-score of 0.92 achieved when combining all available features. Our findings suggest that for executables, features such as capabilities, and string artifacts can serve as strong indicators of similarity or dissimilarity between samples. Notably, incorporating string features significantly boosted the F1-score for executable clustering (from 70% to 89%), whereas the impact on document clustering was less pronounced (from 79% to 91%). This is likely because printable strings in executables are more informative than hierarchically structured formats of documents and PDFs. Šrndić et al. [99] also highlight this observation in their work on PDF and Adobe Flash files.

9 Qualitative Case Studies

In this section, we evaluate ADAPT’s clustering using the group-labeled dataset (discussed in Section 4.1). We focus on two practical use cases for analysts: (1) We discuss the results of ADAPT’s campaign and group attribution for randomly selected clusters from Gamaredon, APT29, and Lazarus, and investigate reasons for misclustering (see Section 9.1). (2) We discuss ADAPT’s attribution of unlabeled samples to known threat groups (see Section 9.2).

9.1 Attributing Labeled Samples

Insights on Threat Campaigns: Gamaredon’s 2017 & 2022 Campaigns. Table 6 (in the Appendix) shows distinct threat campaigns perpetrated by Gamaredon, a suspected Russian cyber espionage threat group [73]. The campaigns occurred in 2017 and 2022, and using ADAPT Intra-Clustering, we successfully grouped the samples belonging to these campaigns into the Clusters #C1 and #C2, respectively.

Unit42 reported the Cluster #C1 samples as a part of the 2017 campaign, which primarily targeted individuals involved in the Ukrainian military and national security establishment [44]. The threat actors distributed custom-developed Windows malware that could download additional payloads, which were distributed as password-protected self-extracting Zip-archive (.SFX) files. These SFX files wrote a batch script to disk and installed a remote access trojan. ADAPT clustered the samples together based on the following shared similarities: the privilege escalation capability via `AdjustTokenPrivileges`, the unique linker `MSVC_2008_linker`, a distinct string formatting pattern used in naming temporary files (`%s.%d.tmp`), and the use of `RIPEMD160` for file encryption.

Unit42 reported the Cluster #C2 samples in January 2022 after observing Gamaredon’s attempt to compromise a Western government entity in Ukraine following the Russia-Ukraine conflict [107]. This campaign uses a custom Windows Remote Access Trojan (RAT) with anti-detection features to evade antiviruses and sandbox environments. Further, it is capable of downloading and executing files, capturing screenshots, and running arbitrary commands on compromised systems. Although the samples in this campaign evolved over time, ADAPT successfully found unique elements that persisted through changes, such as the use of a shared linker and compiler (`MSVC_2005_linker` and `msvc_uv_55`) and the use of `GetKeyState` for keylogging. Additionally, the unique batch script pattern `7ZSfx%03x.cmd` and the embedded icons with Russian-language naming convention (`ru-ru`) helped ADAPT to cluster the samples.

Takeaway 1: ADAPT’s feature extraction improves basic static analysis for threat analysts, giving quick insights into related samples. Additionally, ADAPT’s Intra-Clustering model can identify samples from different campaigns, helping analysts build and track adversary profiles.

Insights on Threat Groups: APT29 & Lazarus. Table 7 (in the Appendix) shows a subset of sample hashes associated with APT29. These hashes were exposed in a July 2020 advisory report, revealing the WellMess and WellMail custom malware [74]. This malware targeted COVID-19 vaccine developers in Canada, the United States, and the United Kingdom to steal vaccine-related data. Interestingly, Japan’s Computer Emergency Response Team (CERT) observed the deployment of WellMess within a Japanese organization as early as 2018 [104], indicating the malware’s presence in various campaigns targeting different organizations. Initially, unattributed, further investigations by the NSA and NSCS linked WellMess and WellMail to APT29, Russia’s Foreign Intelligence Service, which is also suspected of orchestrating the SolarWinds campaign [85].

Using group-based linking characteristics to attribute samples across multiple file types, ADAPT successfully clustered these hashes belonging to the same threat group in Cluster #G1, encompassing WellMess and WellMail campaigns dating back to 2017. The samples include both ELF and PE executables, with some of these binaries programmed in Go for cross-compatibility. Inspecting the shared features that caused ADAPT to group the samples together revealed that all the samples used similar Golang module file paths, such as `/home/ubuntu/GoProject/src/bot/botlib.Work`, `/usr/local/go/src/net/fd_mutex.go`, and `/golang.org/x/crypto/curve25519.freeze`. This suggests the use of mutexes and the `curve25519` crypto library. Additionally, we identified similar MD5 hashes and matching regex patterns embedded in the string content of the samples, including `1DecemberDuployanDuration` `Ethiopic` and `15625AdjustTokenPrivilegesAlaskan`.

In another example, Table 8 (in the Appendix) shows two MachO samples associated with the North Korean threat group, Lazarus, namely `JMTTrade` and `CelasTradePro`. These malware samples, disguised as legitimate cryptocurrency trading applications, have been used in separate campaigns by Lazarus to target both Windows and Mac systems since at least 2018 [19]. ADAPT effectively grouped these samples together in Cluster #G2 due to a recurring Bitcoin pattern embedded within their string content. Notably, the associated Bitcoin wallet addresses remain active and have received a total of \$179,442 in funds [12]. Furthermore, inspecting the common linking feature set we identified similar URLs and shared email addresses, such as `knzg75@jmttrading.org` and `altan73@jmttrading.org`, that facilitated the clustering of these samples. Additionally, the samples share a distinct certificate issuer organization, such as `WoTrus CA Limited`, as observed from the infrastructure feature set. In February 2021, the United States government charged three individuals in connection with this attack, which resulted in a \$1.3 billion theft. These charges were based on infrastructure similarities and online accounts observed in prior campaigns [25].

These examples demonstrate the effectiveness of using pattern-based features, such as system file paths, unique file names, email addresses, Bitcoin identifiers, certificate authorities, and domain names or URLs, to identify threat actor indicators across campaigns. While these distinct signatures have traditionally been used by threat analysts in manual analysis, ADAPT streamlines and automates the process of extracting and clustering these key patterns.

Takeaway 2: ADAPT’s Inter-Clustering model uses distinct patterns and infrastructural details to identify threat actor signatures. This capability expedites the process of connecting attacks carried out by the same threat actor.

Clustering Issues. Although effective in most cases, ADAPT’s clustering might not be ideal in specific scenarios. For instance, while analyzing document clusters we identified an outlier: A document, linked to ‘Operation Dream Job’ (espionage using fake defense jobs [72]), was not grouped within that category. While this sample exhibited some differences compared to others in the campaign, a unique characteristic was the use of similar themes within the document content (e.g., job descriptions, industry jargon)

and a Boeing image on the first page. This demonstrates that including image extraction in the document processing pipeline can improve the ability of ADAPT to cluster similar documents based on visual features.

For executables, obfuscation can hinder clustering. For example, the ‘Andromeda’ campaign [72] used known packers to obfuscate malware. Older samples (first observed in 2013) employed NSIS-based packers, while newer versions (observed in 2022) were unpacked. Our current clustering process grouped the packed samples together, but the unpacked version ended up in a separate cluster, highlighting the need for extracting obfuscated content for efficient clustering. We discuss the issue of packing and obfuscation, including a cursory study on how widespread these techniques are across our dataset, in Section 10.

In some cases, the performance of ADAPT’s clustering for group attribution was limited by a lack of distinctive threat group characteristics. For instance, some samples from Gamaredon, Wizard-Spider, and APT29 were grouped together due to the absence of infrastructure-based indicators and they accessed common file paths (e.g., `C:\Windows\System32\cmd.exe` or `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`). As discussed in Section 6, we successfully extracted Infrastructure (INF) features for 38.22% of the samples. Consequently, the clustering relied on Pattern-based features (PAT), which may not be sufficient for accurate group attribution. However, as detailed in Section 7, these cases can be identified through feature analysis and can be avoided using threshold-based clustering techniques.

Finally, we also recognize limitations in ADAPT’s clustering, particularly when APT groups use publicly available offensive security tools, such as Cobalt Strike, and employ shared infrastructure. This can result in campaigns from unrelated groups being clustered together. Possible remediations include disassembled code analysis to identify code-level differences, and examining tool-related characteristics such as Cobalt Strike license numbers. Another avenue for improvement is incorporating features from dynamic and behavioral analysis, yet with the caveat of dealing with environment-sensitive samples [53].

9.2 Attributing Unlabeled Samples

ADAPT’s group attribution identified nine out of the 44 unlabeled samples in the group-labeled dataset (see Section 4.1) as belonging to specific APT groups. To verify these attributions, we compared the samples to all publicly available information, including community comments on VirusTotal. Four of the samples had comments linking them to the APT group that ADAPT clustered them in, even though the samples were not initially labeled with those groups because there were no public reports of attribution. These samples were clustered based on shared URLs, IPs, similar country codes, and ASNs. Additionally, one of the samples was clustered with three other samples from APT40 (Chinese state-sponsored cyber espionage group) based on similar BGP prefixes and ASNs. The only public information about the sample’s attribution was that it was linked to a Chinese state actor. Another sample was clustered with four other samples from APT10, but the community information linked it to APT3, which are both Chinese threat actors. Finally, three samples were clustered with known APT groups that

had never been publicly attributed to any threat actor. The first sample was grouped with another APT3 sample due to a Bitcoin address pattern. The second sample was linked with six APT28 samples based on string patterns. The third sample was linked to two Kimsuky document samples because of distinct file paths and shared Korean domains in the URLs.

We acknowledge that other organizations might have information about these samples and can validate our attributions. To help the community further analyze and verify our findings, we list ADAPT’s clustering results and the samples’ hashes in Table 9 (in the Appendix), as well as release their features and relevant information as part of our artifact.

10 Discussion and Future Work

Long-lived APT campaigns pose a significant threat due to their stealthy nature. Traditional methods struggle, requiring the tracking of numerous, chronologically linked events over extended periods [3, 14]. Our approach, ADAPT, bypasses this limitation by analyzing inherent features within suspicious or malicious files, independent of their execution sequence. As detailed in our background and motivation (see Section 2), APT groups employ variations in attack vectors and sample modifications throughout a campaign. By identifying shared techniques and capabilities from these heterogeneous artifacts, ADAPT helps analysts rapidly attribute the attack, prioritize analysis, and streamline investigations – even when the attack unfolds asynchronously. Still, we acknowledge open challenges and avenues for future work:

Packing and Obfuscation. Following the discussion in Section 9 we explored the application of common obfuscation techniques on our group-labeled dataset comprising 6,134 samples. Based on previous experiments and recognizing the limitations of existing packer detection tools [1, 64], we used a combination of Manalyzer, Detect it Easy, and Yara rules. These tools collectively flagged 222 samples (3.61%) from the entire dataset, indicating the presence of common obfuscation techniques in those samples.

Content Extraction from Heterogeneous Files. The goal of ADAPT is to use static features for efficient clustering of diverse files. To improve the attribution results, we aim to develop techniques for identifying and extracting obfuscated and embedded content (e.g., PDFs) from executables. Additionally, future efforts will focus on incorporating robust malicious content extraction from heterogeneous file formats. Our orthogonal study on malicious documents [93] highlighted the complexity of non-binary files, particularly Microsoft Office documents (Word, Excel, PowerPoint) and RTFs, and their widespread use among sophisticated attackers. Our study revealed limitations in current document analysis approaches, such as the inability to correctly identify file formats and manage emerging file types like OneNote.

Adversarial Manipulation and Evasion. We acknowledge that certain features within ADAPT may be susceptible to manipulation by adversaries, potentially enabling them to evade accurate attribution. However, it is important to note that, to the best of our knowledge, ADAPT is the first system that explores the feasibility of performing both campaign and group attribution using lightweight features from diverse threat sources. Our approach leverages domain expertise to identify features useful for attribution.

Table 5: Overview of Prior Studies on APT Detection and Attribution. Our approach is the only one that handles executables (📄) and documents (📄) and performs both threat group and campaign attribution, unlike the other malware-based attribution systems. Note that the limited availability of artifacts (code or data) from prior studies limits our ability to perform benchmarks against them.

Approach	Dataset	Samples	Groups	Campaigns	Artifacts
<i>APT detection using alert correlation SIEM/EDR</i>	Synthetic data [31]	–		●	
	Third-party Enterprise Dataset [92]	–	●	●	
<i>APT detection through data provenance</i>	DARPA TC and enterprise logs [34, 42, 51, 55]	–		●	●
<i>APT attribution with knowledge graphs</i>	1,041 OSCTI reports [84]	–	●		
<i>APT attribution based on malware samples</i>	1,569 samples, 16 APT groups [110]	📄	●		
	864 samples, 5 APT groups [33]	📄	●		
	3,200 samples, 2 APT groups [89]	📄	●		
	287 samples, 7 APT campaigns [69]	📄		●	📄
Our Approach: ADAPT	6,134 samples, 92 APT groups 230 samples, 17 APT groups, 22 APT campaigns	📄📄	●	●	●

Following this initial selection, as a part of future work, we aim to rigorously evaluate the robustness of these features against adversarial manipulation, particularly the use of “false flags” [10]. False flags differ from traditional evasion techniques that aim to bypass detection (e.g., benign-appearing actions) or hinder analysis (e.g., anti-analysis tricks). Instead, false flags allow attackers to mask their true identity and deflect attribution towards another nation state. This is the closest adversarial manipulation technique seen in the wild among APTs. To this extent, we identified samples known to exhibit code reuse across multiple threat actors. In particular, we analyzed a set of malware samples exhibiting code reuse across disparate threat actors [101]. These samples were initially misattributed by analysts to a single threat actor due to shared VBA macro code (embedded code in documents). Using ADAPT, we were able to correctly cluster the samples into distinct groups. This differentiation was achieved by analyzing unique identifiers within the samples, such as file paths for destination folders and scheduled task creation methods present in a specific subset of samples.

Concept Drift. Concept drift refers to the phenomenon where the underlying distribution of data changes over time. In our context, this could manifest as new APT campaigns are emerging, potentially leading to changes in how samples are grouped during clustering. Future work will focus on investigating concept drift’s impact on ADAPT’s *unsupervised* setting using rigorous experiments. This would necessitate using a timestamped dataset to track how the underlying data distribution evolves. Specifically, we are interested in observing if new binaries consistently fall within existing clusters or form entirely new clusters as the attack landscape changes.

Representative Dataset. The absence of ground truth datasets that encompass multiple threat groups and include heterogeneous file types remains a significant open research challenge. Furthermore, we did not encounter any dataset with threat campaign labels, prompting us to curate our own dataset (see Section 4), which we provide as part of our artifact. Still, the scale of our dataset is limited, with, on the one hand, APT samples inherently being less widespread than generic malware and the effort involved in manually (re-)labeling the samples.

Furthermore, Arp et al. [8] showed that data sampling biases and data snooping can invalidate the results of machine learning models in security applications. We mitigate these risks by using

real-world malicious files from trusted sources instead of synthetic datasets. However, sampling bias is still possible due to the over-representation of certain groups and file types. Additionally, data snooping is less straightforward in unsupervised clustering than in supervised learning, so we exercise caution in data handling. We use the reference dataset only to evaluate clustering results based on the chosen algorithm and generalized parameters developed during the clustering phase.

11 Related Work

Table 5 shows a summary of the most closely related prior work on APTs. In the following, we discuss related APT datasets, and prior approaches on APT detection and attribution.

APT Datasets. Gray et al. [32] developed a promising APT dataset comprising 17,513 APT samples belonging to 275 APT groups by mining threat reports. However, their dataset only included executable samples (PE, and ELF files) and lacked other file types, particularly documents. Similarly, Laurenza et al. [48] curated a dataset comprising exclusively binary samples, whereas the dataset from cyber-research [22] contains APT samples belonging to only 12 APT groups. Our dataset includes a wide range of file types from 92 APT groups, overcoming the limitations of previous datasets.

APT Detection. Existing research in APT detection leverages alert correlation to identify anomalous behaviors or APT footprints. Ghafir et al. [31] proposed MLAPT, a machine learning system for APT detection using network traffic data, while Sachinananda et al. [92] focus on correlating security alerts from various sources, such as Intrusion Detection and Prevention Systems (IDS/IPS), Endpoint Detection and Response (EDR), and Security Information and Event Management (SIEM) to cluster alerts associated with the same APT attack scenario. Provenance graphs have also emerged as a state-of-the-art approach in APT detection. ANUBIS [7] leverages provenance graphs to capture causality and detect APTs. Similarly, APThunter [55], Unicorn [34], NODLINK [51], and MAGIC [42] focus on provenance-based anomaly detection using audit logs. These approaches, however, require raw log access, suffer from dependency explosion issues, and present challenges in reconstructing complex APT attack causality [37]. Existing research focuses on APT detection and hunting, leaving a gap for a comprehensive framework in APT sample correlation and attribution. We address

this distinct yet complementary aspect. Our work leverages malicious artifacts to facilitate correlation, aiding investigations even without a complete understanding of the attack causality chain.

APT Attribution. Marquis-Boire et al. [65] manually extracted static features specific to APT malware, such as C&C infrastructure, string constants, and data exfiltration methods, to link executables from the same authors. Rosenberg et al. [89] propose deep learning for APT group attribution using sandbox analysis reports of PE binaries. While they perform classification between Chinese and Russian APT groups, specific groups remain undisclosed. Wang et al. [110] explore string and code features with random forest and DNN classifiers for APT malware attribution across 16 APT groups (1,569 samples). Han et al. [33] use dynamic API sequences for APT malware detection and group identification using 864 APT samples. However, limited public availability of the system and dataset, along with potential ground truth issues, hinder the broader evaluation of these approaches. Additionally, Mirzaei et al. [69] propose Scrutinizer, a system for detecting code reuse in PE malware binaries via function-level decompiled code similarity analysis. Through manual verification of samples, they identified 12 previously unknown APT-linked samples. However, Scrutinizer’s reliance on a custom sandbox environment for intermediate results limits reproducibility and an unlabeled dataset of hashes hinders rigorous benchmarking. Finally, Ren et al. [84] propose a cybersecurity knowledge graph model for APT group attribution leveraging OSCTI information.

While the problem of attribution for non-APT malware has been studied extensively [5, 16, 32, 49, 81, 90], these related approaches focus on identifying the author of a binary file and extracting their stylistic features. Another line of related work on commodity malware focuses on so-called lineage [35, 41, 52], i.e., identifying the evolution of and relationships between malware families and variants. In contrast, our research looks at attribution more broadly, and extracts features based on the tactics and techniques of the APT group responsible for the attack and the campaign they are executing. ADAPT advances the state-of-the-art in APT malware attribution by addressing its unique challenges. First, we establish a comprehensive understanding of the APT landscape and its complexities through practical case studies. This includes recognizing the distinct characteristics of APTs, such as low-and-slow tactics and multi-stage attacks involving heterogeneous artifacts. Existing studies primarily focus on PE binaries, while ADAPT performs attribution for executables and the most common initial attack vectors [99], including document file formats. Finally, existing solutions either perform campaign-level or group-level attribution. ADAPT automates both, allowing for a more systematic attribution process. Furthermore, to encourage future research and facilitate reproducibility, we open-source both our dataset, and source code. We aim for transparency and allow researchers to benchmark their approaches against our results.

12 Conclusion

Unlike conventional malware threats, APTs are technically sophisticated adversaries conducting well-organized, stealthy, and repeated campaigns targeting a wide range of organizations. In this paper, we introduce ADAPT, an automated attribution approach that provides insight into the adversary’s tactics and identity by performing APT

campaign and group clustering. ADAPT’s two-tiered approach offers a solution to the challenges of human attribution in the face of evolving and strategic threat campaigns, the use of different file types in attacks, and the collaboration among threat groups that complicates the attribution process. ADAPT clusters executable and document samples by analyzing malicious characteristics. It also uses linking features to identify group traits and signatures, helping connect samples from different campaigns to the same threat group. Practical case studies on real-world APTs and the association of unattributed samples to known APT groups demonstrate how ADAPT simplifies the attribution process, empowering security practitioners with automated tools to assess and compare attribution claims. We envision our work involving the collection of campaign- and group-labeled APT datasets, automated feature extraction for diverse file types, and the use of clustering techniques for attribution analysis as a source of inspiration for future research.

Acknowledgements

We thank the anonymous reviewers for their valuable insights for improving the paper. We would also like to thank Godwin Attigah for his help in setting up the resources for downloading malicious samples, Matthias Glinzner for his assistance with the ground-truth labeling of the ADAPT dataset, and Daniel Arp for his valuable feedback on the organization of the paper. We also thank Censys for providing research access to their data, VirusTotal for providing access to the samples and associated file reports, and AlienVault for their access to the threat intelligence feed.

This material is based on research supported by the Vienna Science and Technology Fund (WWTF) and the City of Vienna [10.47379/ICT19056], the SecInt Doctoral College at TU Wien, and SBA Research (SBA-K1), a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. The COMET Programme is managed by FFG. This work has also been partially supported by Google ASPIRE Awards, as well as the Recovery, Transformation, and Resilience Plan funded by the European Union (Next Generation).

References

- [1] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. 2020. When Malware is Packing Heat; Limits of Machine Learning Classifiers based on Static Analysis Features. In *Proc. of the 27th Network and Distributed System Security Symposium (NDSS)*.
- [2] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto. 2016. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. In *Proc. of the 6th ACM Conference on Data and Application Security and Privacy (CODASPY)*.
- [3] Olusola Akinrolabu, Ioannis Agrafiotis, and Arnau Erola. 2018. The Challenge of Detecting Sophisticated Attacks: Insights from SOC Analysts. In *Proc. of the 13th International Conference on Availability, Reliability and Security (ARES)*.
- [4] AlienVault. 2022. Open Threat Exchange. <https://otx.alienvault.com/>.
- [5] Saeed Alrabaa, Paria Shirani, Mourad Debbabi, and Lingyu Wang. 2016. On the Feasibility of Malware Authorship Attribution. In *Proc. of the 9th International Symposium on Foundations and Practice of Security (FPS)*.
- [6] Hyrum S. Anderson and Phil Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. (2018). arXiv: 1804.04637 [cs.CR].
- [7] Md Monwar Anjum, Shahrear Iqbal, and Benoit Hamelin. 2022. ANUBIS: A Provenance Graph-Based Framework for Advanced Persistent Threat Detection. In *Proc. of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC)*.

- [8] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don't of Machine Learning in Computer Security. In *Proc. of the 31st USENIX Security Symposium (USENIX Security)*.
- [9] Michael Barnhart, Austin Larsen, Jeff Johnson, Taylor Long, Michelle Cantos, and Adrian Hernandez. 2023. Assessed Cyber Structure and Alignments of North Korea in 2023. (Oct. 10, 2023). <https://www.mandiant.com/resources/blog/north-korea-cyber-structure-alignment-2023>.
- [10] Brian Bartholomew and Juan Andres Guerrero-Saade. 2016. Wave Your False Flags! Deception Tactics Muddying Attribution in Targeted Attacks. In *Virus Bulletin Conference*.
- [11] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. 2009. Scalable, Behavior-based Malware Clustering. In *Proc. of the 16th Network and Distributed System Security Symposium (NDSS)*.
- [12] Blockchain.com. 2016. Bitcoin Address. <https://www.blockchain.com/explore/r/addresses/btc/1QLDYeyeo8c6CFHdcEB5yBjQw6pcRiTdN5>.
- [13] Marcus Botacin, Hojjat Aghakhani, Stefano Ortolani, Christopher Kruegel, Giovanni Vigna, Daniela Oliveira, Paulo Lício De Geus, and André Grégio. 2021. One Size Does Not Fit All: A Longitudinal Analysis of Brazilian Financial Malware. *ACM Transactions on Privacy and Security (TOPS)*, 24, 2.
- [14] Guillaume Brogi and Valerie Viet Triem Tong. 2016. TerminAPTor: Highlighting Advanced Persistent Threats through Information Flow Tracking. In *Proc. of the 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*.
- [15] Rufus Brown, Van Ta, Douglas Bienstock, Geoff Ackerman, and John Wolfram. 2022. Does This Look Infected? A Summary of APT41 Targeting U.S. State Governments. (Mar. 8, 2022). <https://www.mandiant.com/resources/blog/apt41-us-state-governments>.
- [16] Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. 2015. De-anonymizing Programmers via Code Stylometry. In *Proc. of the 24th USENIX Security Symposium (USENIX Security)*.
- [17] Microsoft Security Response Center. 2020. Tracking the Cross-Domain Solorigate Attack from Endpoint to the Cloud. (Dec. 28, 2020). <https://www.microsoft.com/en-us/security/blog/2020/12/28/using-microsoft-365-defender-to-coordinate-protection-against-solorigate/>.
- [18] Tencent Security Threat Intelligence Center. 2019. Cyber Warfare in the Shadow of the India-Pakistan War - A Summary of Recent Indo-Pakistani APT Attack Activities. (Sept. 9, 2019). <https://mp.weixin.qq.com/s/pj-rnzB7VMZ0feM2X0zrHA>.
- [19] CISA. 2021. AppleJeus: JMT Trading. (Apr. 15, 2021). <https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-048a>.
- [20] Eric Cole. 2012. *Advanced Persistent Threat: Understanding the Danger and How to Protect your Organization*. Syngress Publishing.
- [21] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. 2018. Understanding Linux Malware. In *Proc. of the 39th IEEE Symposium on Security & Privacy (S&P)*.
- [22] cyber-research. 2019. APT Malware Dataset. (July 16, 2019). <https://github.com/cyber-research/APTMalware>.
- [23] Cyble. 2020. Sidewinder APT Targets with Futuristic Tactics and Techniques. (Sept. 26, 2020). <https://blog.cyble.com/2020/09/26/sidewinder-apt-targets-with-futuristic-tactics-and-techniques/>.
- [24] Cyware. 2020. DoNot Team APT Updates its Malware Arsenal. (Aug. 12, 2020). <https://cyware.com/news/donot-team-apt-updates-its-malware-arsenal-a5a76e92>.
- [25] DOJ. 2021. DPRK Hacking Indictment. (Feb. 17, 2021). https://www.justice.gov/d9/press-releases/attachments/2021/02/17/dprk_hacking_-_indictment_1_0.pdf.
- [26] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
- [27] Hugging Face. 2022. all-MiniLM-L12-v2. <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2>.
- [28] Hugging Face. 2022. Multi-qa-mpnet-base-dot-v1. <https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1>.
- [29] FBI. 2018. APT 10 Group. (Dec. 17, 2018). <https://www.fbi.gov/wanted/cyber/apt-10-group>.
- [30] FBI. 2020. APT 41 Group. (Aug. 11, 2020). <https://www.fbi.gov/wanted/cyber/apt-41-group>.
- [31] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J Aparicio-Navarro. 2018. Detection of Advanced Persistent Threat using Machine-Learning Correlation Analysis. *Future Generation Computer Systems*, 89.
- [32] Jason Gray, Daniele Sgandurra, Lorenzo Cavallaro, and Jorge Blasco. 2024. Identifying Authorship in Malicious Binaries: Features, Challenges & Datasets. *ACM Computing Surveys (CSUR)*, 56, 8.
- [33] Weijie Han, Jingfeng Xue, Yong Wang, Fuquan Zhang, and Xianwei Gao. 2021. APTMalInsight: Identify and Cognize APT Malware Based on System Call Information and Ontology Knowledge Framework. *Information Sciences*, 546.
- [34] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. In *Proc. of the 27th Network and Distributed System Security Symposium (NDSS)*.
- [35] Irfan Ul Haq, Sergio Chica, Juan Caballero, and Somesh Jha. 2018. Malware Lineage in the Wild. *Computers & Security*, 78.
- [36] Hawkeye. 2022. The Evolution of Sidewinder APT and their Modus Operandi. <https://www.hawkeye.io/2022/12/the-evolution-of-sidewinder-apt-and-their-modus-operandi/>.
- [37] Md Nahid Hossain, Sanaz Sheikh, and R. Sekar. 2020. Combating Dependence Explosion in Forensic Analysis using Alternative Tag Propagation Semantics. In *Proc. of the 41st IEEE Symposium on Security & Privacy (S&P)*.
- [38] The White House. 2021. Fact Sheet: Imposing Costs for Harmful Foreign Activities by the Russian Government. (Apr. 15, 2021). <https://www.whitehouse.gov/briefing-room/statements-releases/2021/04/15/fact-sheet-imposing-costs-for-harmful-foreign-activities-by-the-russian-government/>.
- [39] Zheng Hu, Jiaojiao Zhang, and Yun Ge. 2021. Handling Vanishing Gradient Problem using Artificial Derivative. *IEEE Access*, 9.
- [40] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data Clustering: A Review. *ACM Computing Surveys (CSUR)*, 31, 3.
- [41] Jiyong Jang, Maverick Woo, and David Brumley. 2013. Towards Automatic Software Lineage Inference. In *Proc. of the 22nd USENIX Security Symposium (USENIX Security)*.
- [42] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2023. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In *Proc. of the 32nd USENIX Security Symposium (USENIX Security)*.
- [43] Vaibhavi Kalgutkar, Natalia Stakhanova, Paul Cook, and Alina Matyukhina. 2018. Android Authorship Attribution through String Analysis. In *Proc. of the 13th International Conference on Availability, Reliability and Security (ARES)*.
- [44] Anthony Kasza and Dominik Reiche. 2017. The Gamaredon Group Toolset Evolution. (Feb. 27, 2017). <https://unit42.paloaltonetworks.com/unit-42-title-gamaredon-group-toolset-evolution/>.
- [45] Sangwoo Kim, Seokmyung Hong, Jaesang Oh, and Heejo Lee. 2018. Obfuscated VBA Macro Detection using Machine Learning. In *Proc. of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
- [46] Alien Labs. 2021. A Global Perspective of the SideWinder APT. (Jan. 13, 2021). <https://cdn-cybersecurity.att.com/docs/global-perspective-of-the-sidewinder-apt.pdf>.
- [47] Philippe Lagadec. 2022. Oletools - Python Tools to Analyze OLE and MS Office Files (version 0.60.1). (May 9, 2022). <https://github.com/decalage2/oletools>.
- [48] Giuseppe Laurenza and Riccardo Lazzarotti. 2019. dAPTaset: A Comprehensive Mapping of APT-Related Data. In *Proc. of the International Workshop on Security for Financial Critical Infrastructures and Services (FINSEC)*.
- [49] Robert Layton and Ahmad Azab. 2014. Authorship Analysis of the Zeus Botnet Source Code. In *Proc. of the 5th Cybercrime and Trustworthy Computing Conference (CTC)*.
- [50] Clement Lecigne and Maddie Stone. 2023. Active North Korean Campaign Targeting Security Researchers. (Sept. 7, 2023). <https://blog.google/threat-analysis-group/active-north-korean-campaign-targeting-security-researchers/>.
- [51] Shaofei Li, Feng Dong, Xusheng Xiao, Haoyu Wang, Fei Shao, Jiedong Chen, Yao Guo, Xiangqun Chen, and Ding Li. 2024. NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation. In *Proc. of the 31st Network and Distributed System Security Symposium (NDSS)*.
- [52] Martina Lindorfer, Alessandro Di Federico, Federico Maggi, Paolo Milani Comparetti, and Stefano Zanero. 2012. Lines of Malicious Code: Insights Into the Malicious Software Industry. In *Proc. of the 28th Annual Computer Security Applications Conference (ACSAC)*.
- [53] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. 2011. Detecting Environment-Sensitive Malware. In *Proc. of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID)*.
- [54] Martina Lindorfer, Bernhard Miller, Matthias Neugschwandtner, and Christian Platzer. 2013. Take a Bite - Finding the Worm in the Apple. In *Proc. of the 9th International Conference on Information, Communications and Signal Processing (ICIS)*.
- [55] Moustafa Mahmoud, Mohammad Mannan, and Amr Youssef. 2022. APThunter: Detecting Advanced Persistent Threats in Early Stages. *Digital Threats: Research and Practice*, 4.
- [56] Malcat. 2024. Binary Analysis Software (version 0.8.3). <https://malcat.fr/>.
- [57] Malpedia. 2024. Lazarus Group. (Mar. 2024). https://malpedia.caad.fkie.fraunhofer.de/actor/lazarus_group.
- [58] Mandiant. 2021. APT1: Exposing One of China's Cyber Espionage Units. (Dec. 30, 2021). <https://www.mandiant.com/resources/apt1-exposing-one-of-chinas-cyber-espionage-units>.

- [59] Mandiant. 2022. Advanced Persistent Threats (APTs). <https://www.mandiant.com/resources/insights/apt-groups>. (2022).
- [60] Mandiant. 2022. APT42: Crooked Charms, Cons and Compromises. (Aug. 12, 2022). <https://www.mandiant.com/media/17826>.
- [61] Mandiant. 2022. Supply Chain Analysis: From Quartermaster to Sunshop. (Jan. 20, 2022). <https://www.mandiant.com/resources/supply-chain-analysis-from-quartermaster-to-sunshop>.
- [62] Mandiant. 2023. FLOSS - FLARE Obfuscated String Solver (v2.2.0). (Dec. 12, 2023). <https://github.com/mandiant/flare-floss>.
- [63] Mandiant. 2024. Uncategorized (UNC) Threat Groups. (Mar. 2024). <https://www.mandiant.com/resources/insights/uncategorized-unc-threat-groups>.
- [64] Alessandro Mantovani, Simone Aonzo, Xabier Ugarte-Pedrero, Alessio Merlo, and Davide Balzarotti. 2020. Prevalence and Impact of Low-Entropy Packing Schemes in the Malware Ecosystem. In *Proc. of the 27th Network and Distributed System Security Symposium (NDSS)*.
- [65] Morgan Marquis-Boire, Marion Marschalek, and Claudio Guarnieri. 2015. Big Game Hunting: The Peculiarities in Nation-State Malware Research. In *BlackHat USA*.
- [66] Francesco Meloni, Alessandro Sanna, Davide Maiorca, and Giorgio Giacinto. 2022. Effective Call Graph Fingerprinting for the Analysis and Classification of Windows Malware. In *Proc. of the 19th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.
- [67] Microsoft. 2023. Manage Exclusions for Microsoft Defender. (July 8, 2023). <https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/defender-endpoint-antivirus-exclusions>.
- [68] Mamoru Mimura and Taro Ohminami. 2019. Towards Efficient Detection of Malicious VBA Macros with LSI. In *Proc. of the 14th International Workshop on Security (IWSEC)*.
- [69] Omid Mirzaei, Roman Vasilenko, Engin Kirda, Long Lu, and Amin Kharraz. 2021. Scrutinizer: Detecting Code Reuse in Malware via Decompilation and Machine Learning. In *Proc. of the 18th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.
- [70] MITRE. 2020. Sidewinder. <https://attack.mitre.org/groups/G0121/>.
- [71] MITRE. 2021. Transparent Tribe. <https://attack.mitre.org/groups/G0134/>.
- [72] MITRE. 2022. MITRE Campaigns. <https://attack.mitre.org/campaigns/>.
- [73] MITRE. 2023. MITRE Groups. <https://attack.mitre.org/groups/>.
- [74] NCSC. 2020. Advisory: APT29 Targets COVID-19 Vaccine Development. (July 16, 2020). https://media.defense.gov/2020/Jul/16/2002457639/-1/-1/0/NCSC_APT29_ADVISORY-QUAD-OFFICIAL-20200709-1810.PDF.
- [75] NIST. 2017. CVE-2017-11882. (Nov. 14, 2017). <https://nvd.nist.gov/vuln/detail/CVE-2017-11882>.
- [76] NIST. 2024. Advanced Persistent Threats. <https://csrc.nist.gov/topics/security-and-privacy/risk-management/threats/advanced-persistent-threats>.
- [77] NIST. 2024. Threat Actor. https://csrc.nist.gov/glossary/term/threat_actor.
- [78] NIST. 2024. Threat Scenario. https://csrc.nist.gov/glossary/term/threat_scenario.
- [79] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Courville, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12.
- [80] Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral Clustering of Http-Based Malware and Signature Generation using Malicious Network Traces. In *Proc. of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*.
- [81] Avi Pfeffer, Catherine Call, John Chamberlain, Lee Kellogg, Jacob Ouellette, Terry Patten, Greg Zacharias, Arun Lakhotia, Suresh Golconda, John Bay, Robert Hall, and Daniel Scofield. 2012. Malware Analysis and Attribution using Genetic Information. In *Proc. of the 7th International Conference on Malicious and Unwanted Software (MALWARE)*.
- [82] PyPi. 2022. Python Magic. (June 7, 2022). <https://pypi.org/project/python-magic/>.
- [83] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proc. of the 9th International Joint Conference on Natural Language Processing (IJCNLP)*.
- [84] Yitong Ren, Yanjun Xiao, Yinghai Zhou, Zhiyong Zhang, and Zhihong Tian. 2022. CSKG4APT: A Cybersecurity Knowledge Graph for Advanced Persistent Threat Organization Attribution. *IEEE Transactions on Knowledge and Data Engineering*, 35.
- [85] Reuters. 2021. SolarWinds Hack was Largest and Most Sophisticated Attack Ever: Microsoft President. (Feb. 15, 2021). <https://www.reuters.com/article/us-cyber-solarwinds-microsoft-idUSKBN2AF03R>.
- [86] Rewterz. 2024. Rewterz Threat Alert - SideWinder APT Group aka Rattlesnake - Active IOCs. (Mar. 8, 2024). <https://www.rewterz.com/rewterz-news/rewterz-threat-alert-sidewinder-apt-group-aka-rattlesnake-active-iocs-2/>.
- [87] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. 2008. Learning and Classification of Malware Behavior. In *Proc. of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*.
- [88] Md Omar Faruk Rokon, Risul Islam, Ahmad Darki, Evangelos E Papalexakis, and Michalis Faloutsos. 2020. SourceFinder: Finding Malware Source-Code from Publicly Available Repositories in GitHub. In *Proc. of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*.
- [89] Ishai Rosenberg, Guillaume Sicard, and Eli Omid David. 2017. DeepAPT: Nation-State APT Attribution Using End-to-End Deep Neural Networks. In *Proc. of the 26th International Conference on Artificial Neural Networks (ICANN)*.
- [90] Nathan Rosenblum, Xiaojin Zhu, and Barton P Miller. 2011. Who Wrote this Code? Identifying the Authors of Program Binaries. In *Proc. of the 16th European Symposium on Research in Computer Security (ESORICS)*.
- [91] Florian Roth. 2018-03-25. The Newcomer's Guide to Cyber Threat Actor Naming. <https://cyb3rops.medium.com/the-newcomers-guide-to-cyber-threat-actor-naming-7428e18ee263>.
- [92] Vinay Sachidananda, Rajendra Patil, Akshay Sachdeva, Kwok-Yan Lam, and Liu Yang. 2023. APTer: Towards the Investigation of APT Attribution. In *Proc. of the 6th IEEE Conference on Dependable and Secure Computing (DSC)*.
- [93] Aakanksha Saha, Jorge Blasco, and Martina Lindorfer. 2024. Exploring the Malicious Document Threat Landscape: Towards a Systematic Approach to Detection and Analysis. In *Proc. of the 3rd Workshop on Rethinking Malware Analysis (WoRMA)*.
- [94] scikit-learn. 2024. Hierarchical Clustering. <https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering>.
- [95] scikit-learn. 2024. Silhouette Score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.
- [96] Elastic Security. 2024. Elastic Security Detection Content for Endpoint. <https://github.com/elastic/protections-artifacts>.
- [97] Microsoft Security. 2023. Microsoft Shifts to a New Threat Actor Naming Taxonomy. (Apr. 2023). <https://www.microsoft.com/en-us/security/blog/2023/04/18/microsoft-shifts-to-a-new-threat-actor-naming-taxonomy/>.
- [98] Krzysztof Siwek and Stanislaw Osowski. 2017. Autoencoder versus PCA in Face Recognition. In *Proc. of the 18th International Conference on Computational Problems of Electrical Engineering (CPEE)*.
- [99] Nedim Šrđić and Pavel Laskov. 2016. Hidost: A Static Machine-Learning-Based Detector of Malicious Files. *EURASIP Journal on Information Security*.
- [100] Ting Su and Jennifer Dy. 2004. A Deterministic Method for Initializing K-means Clustering. In *Proc. of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*.
- [101] Vanja Svajcer and Vitor Ventura. 2022. What's with the shared VBA code between Transparent Tribe and other threat actors? (Feb. 9, 2022). <https://blog.talosintelligence.com/2022/02/whats-with-shared-vba-code.html>.
- [102] Romain Thomas. 2024. LIEF - Library to Instrument Executable Formats (version 0.13.1). (Feb. 11, 2024). <https://lief.quarkslab.com/>.
- [103] Threatpost. 2020. SideWinder APT Targets Nepal, Afghanistan in Wide-Ranging Spy Campaign. (Dec. 9, 2020). <https://threatpost.com/sidewinder-apt-nepal-afghanistan-spy-campaign/162086/>.
- [104] Shusei Tomonaga. 2018. Malware "WellMess" Targeting Linux and Windows. (July 6, 2018). <https://blogs.jp.cert.or.jp/en/2018/07/malware-wellmes-9b78.html>.
- [105] Trellix. 2022. Trellix Insights: FireEye Red Team Tools Stolen in Cyber Attack. (Aug. 29, 2022). <https://kcm.trellix.com/corporate/index?page=content&id=KB93880>.
- [106] TrendMicro. 2020. SideWinder Uses South Asian Issues for Spear Phishing, Mobile Attacks. (Dec. 9, 2020). https://www.trendmicro.com/de_de/research/2011/sidewinder-leverages-south-asian-territorial-issues-for-spear-ph.html.
- [107] Unit42. 2022. Russia's Gamaredon aka Primitive Bear APT Group Actively Targeting Ukraine. (Feb. 3, 2022). <https://unit42.paloaltonetworks.com/gamaredon-primitive-bear-ukraine-update-2021/>.
- [108] Kelli Vanderlee. 2020. DebUNCing Attribution: How Mandiant Tracks Uncategorized Threat Actors. (Dec. 17, 2020). <https://www.mandiant.com/resources/blog/how-mandiant-tracks-uncategorized-threat-actors>.
- [109] VirusTotal. 2023. VirusTotal. <https://www.virustotal.com/>.
- [110] Qinqin Wang, Hanbing Yan, and Zhihui Han. 2021. Explainable APT Attribution for Malware using NLP Techniques. In *Proc. of the 21st International Conference on Software Quality, Reliability and Security (QRS)*.
- [111] Adam Weidemann. 2021. New Campaign Targeting Security Researchers. (Jan. 25, 2021). <https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/>.
- [112] Yara Rules Project. 2022. Repository of Yara Rules. <https://github.com/Yara-Rules/rules>.
- [113] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M. Blough, Elissa M. Redmiles, and Mustaque Ahamad. 2021. An Inside Look into the Practice of Malware Analysis. In *Proc. of the 28th ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

Table 6: Gamaredon threat campaign analysis. (see case study in Section 9.1)

Campaign	APT Hashes	Campaign Cluster Label
<i>Gamaredon 2017 Campaign</i>	2c5d55619d2f56dc5824a4845334e7804d6d306daac1c23bec6f078f30f1c825	Cluster #C1
	3ef3a06605b462ea31b821eb76b1ea0fdf664e17d010c1d5e57284632f339d4b	Cluster #C1
	4d1a6fe0df9b00f34e3461cb0119224b242c0257b991e8c44a51f0e3304771ea	Cluster #C1
	63fcfab8e9b97d9aec3d6f243003ea3e2bf955523f08e6f1c0d1e28c839ee3d5	Cluster #C1
<i>Gamaredon 2022 Campaign</i>	61e67302a85ff98eabc589572dbf3bf6e1012207d399b9f2b6b38527833e9198	Cluster #C2
	b9dd1e5ec018090b404dd7550d4423ff38ee1f016a5ab214f128544f5b399759	Cluster #C2
	cbe1dbd167bccbf61ee8608092a767ce3fbfb5fe5f6e959848d9a8d9091402fb	Cluster #C2
	3dca96ef38d4b8d1dbb4afed43a22ace93cc3a0a105120d4cf637e6dafe129e9	Cluster #C2

Table 7: APT29 threat group analysis. (see case study in Section 9.1)

Normalized Group Label	APT Hashes	Group Cluster Label
<i>APT29</i>	84b846a42d94431520d3d2d14262f3d3a5d96762e56b0ae471b853d1603ca403	Cluster #G1
	00654dd07721e7551641f90cba832e98c0acb030e2848e5efc0e1752c067ec07	Cluster #G1
	0322c4c2d511f73ab55bf3f43b1b0f152188d7146cc67ff497ad275d9dd1c20f	Cluster #G1
	5ca4a9f6553fea64ad2c724bf71d0fac2b372f9e7ce2200814c98aac647172fb	Cluster #G1
	bec1981e422c1e01c14511d384a33c9bcc66456c1274bbbac073da825a3f537d	Cluster #G1

Table 8: Lazarus threat group analysis. (see case study in Section 9.1)

Normalized Group Label	APT Hashes	Group Cluster Label
<i>Lazarus</i>	7ea6391c11077a0f2633104193ec08617eb6321a32ac30c641f1650c35eed0ea	Cluster #G2
	c0c2239138b9bc659b5bddd8f49fa3f3074b65df8f3a2f639f7c632d2306af70	Cluster #G2

Table 9: Attribution of unlabeled samples. (see discussion in Section 9.2)

Potential Group Label	APT Hashes	Group Cluster Label
<i>APT3</i>	71b201a5a7dfdbe91c0a7783f845b71d066c62014b944f488de5aec6272f907c	Cluster #G3
<i>Transparent Tribe</i>	bff6270b7c6240c394515dc2505bb9f55d7b9df700be1777a8469143f78d0eb6	Cluster #G4
<i>APT40</i>	f659b269f4128588f7a2fa4d6022cc74e508d28eee05c5aff26cc23b7bd1a5	Cluster #G5
<i>APT28</i>	4a9efdfa479c8092fefe182eb7d285de23340e29e6966f1a7302a76503799a2	Cluster #G6
	eae62bb4110bcd00e9d1bcaba9000defcda3d1ab832fa2634d92859d066cb15	Cluster #G7
	b3cee881b2f9d115c98d431b70a75709aade2317a82a0792c15dce2ffa892679	Cluster #G7
<i>APT15</i>	12e1b00af73101cb297387b6ee5035c4cae04211d995ddd233fb375deb492b0a	Cluster #G8
<i>Kimsuky</i>	fa71eee906a7849ba3f4bab74edb577bd1f1f8397ca428591b4a9872ce1f1e9b	Cluster #G9
<i>APT10</i>	df5f1b802d553cddd3b99d1901a87d0d1f42431b366cfb0ed25f465285e38d27	Cluster #GA