TU WIEN Informatics

Improving Enterprise IT Security by Integrating Crowdsourced Offensive Security

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Shpend Kurtishaj
Matrikelnummer 01428454

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Thomas Grechenig

Wien, 20. Mai 2021

Unterschrift Verfasser

Unterschrift Betreuung

Informatics

Improving Enterprise IT Security by Integrating Crowdsourced Offensive Security

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Shpend Kurtishaj
Registration Number 01428454

to the Faculty of Informatics

at the TU Wien

Advisor: Thomas Grechenig

Vienna, 20th May, 2021

Signature Author

Signature Advisor

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

Improving Enterprise IT Security by Integrating Crowdsourced Offensive Security

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Shpend Kurtishaj
Registration Number 01428454

elaborated at the
Institute of Information Systems Engineering
Research Group for Industrial Software
to the Faculty of Informatics
at TU Wien

Advisor: Thomas Grechenig

Vienna, May 24, 2021

Statement by Author

Shpend Kurtishaj
Hasenörlstrasse 67, 1100 Wien

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

I hereby declare that I am the sole author of this thesis, that I have completely indicated all sources and help used, and that all parts of this work – including tables, maps and figures – if taken from other works or from the internet, whether copied literally or by sense, have been labelled including a citation of the source.

(Place, Date)	(Signature of Author)

Kurzfassung

In IT-Sicherheit gewann ein Crowdsourcing-Ansatz zur offensiven Sicherheit (auch als Bug Bounty bekannt) in den letzten Jahren zunehmend an Beliebtheit. In diesem Ansatz lädt eine Organisation externe Sicherheitsforscher ein, um Schwachstellen zu finden und diese zu melden. Im Gegenzug bietet die Organisation eine Belohnung an, üblicherweise in Form von Bezahlung. Dieser Ansatz bietet einige Vorteile gegenüber anderen Sicherheitsmethoden. Erstens wird das Ziel von einer großen Anzahl von Testern mit unterschiedlichen Fähigkeiten getestet, sodass die Wahrscheinlichkeit, Schwachstellen zu erkennen, erhöht wird. Zweitens ist das Kostenmodell attraktiv. Da Belohnungen nur für gültige Schwachstellenberichte angeboten werden, stehen die Kosten in dieser Sicherheitstestmethode eng mit den von ihr generierten Ergebnissen in Zusammenhang.

Trotz der Beliebtheit hat sich ein Crowdsourcing-Ansatz zur offensiven Sicherheit in Großunternehmen nur langsam durchgesetzt. Dies ist auf den Mangel an Informationen und formalen Richtlinien zum Erstellen und Verwalten eines Bug Bounty Programms zurückzuführen. Das Ziel dieser Arbeit ist es diese Lücke zu schließen und einen praktischen Leitfaden für die Integration und Ausführung von Crowdsourced-Sicherheitstest in Großunternehmen bereitzustellen. Für die Integration eines Bug Bounty Programms werden zwei Alternativen vorgestellt: ein institutioneller Ansatz, in dem eine Organisation beschließt ihr Bug Bounty Program selber zu verwalten, und ein Ansatz in dem ein Bug Bounty Dienstleister mit der Verwaltung beauftragt wird. Für beide Optionen bietet diese Arbeit detaillierte Anweisungen zum Starten, Betreiben und Verwalten eines Bug Bounty Programms. Dies umfasst Aspekte wie das Erstellen der Rules of Engagement, das Zusammenstellen des Betriebsteams, das Definieren von Erfolgsmetriken, das Auswählen der richtigen Ziele und das Verarbeiten von Schwachstellenberichten.

Um die Konzepte dieser Arbeit zu testen, wird eine Fallstudie für ein Großprojekt durchgeführt. Experteninterviews werden als zusätzliche Methode zur Bewertung der Leistung dieser Konzepte verwendet. Die Ergebnisse kommen zu dem Schluss, dass Bug Bounties eine wirksame Maßnahme zur Verbesserung der IT-Sicherheit in Großunternehmen darstellt. Die Ergebnisse zeigen auch, dass Crowdsourced-Sicherheitstests nicht herkömmliche Offensive-Sicherheitstestmethoden ersetzten. Stattdessen dienen diese als ergänzende Maßnahme und erhöhen bei gleichzeitiger Verwendung die Wirksamkeit anderer offensiver Sicherheitsmethoden.

Schlüsselwörter

Großunternehmen, Offensive Sicherheit, Crowdsourced-Sicherheitstests, Bug Bounty

Abstract

In IT security, a crowdsourced approach to offensive security has seen increasing popularity in recent years. In this approach, an organization invites external security researchers to find and report vulnerabilities to them. In return, the organization offers some type of reward, commonly in monetary form. This approach provides some advantages compared to other offensive security methods. First, the target is tested by a large number of testers with various skill sets. This increases the exposure of the target and is likely to lead to more vulnerability findings. Second, the return of investment is attractive. Because an organization only pays for valid vulnerability reports, the cost of this offensive security testing methodology is closely connected to the results it generates.

Despite its popularity, crowdsourced offensive security has seen slow adoption amongst enterprise organizations. This is attributed to the lack of information and formal guidelines for creating and managing a bug bounty program. This thesis aims to fill this void and provide a practical guideline for integrating and running crowdsourced offensive security in enterprise IT. Two alternatives for integrating a bug bounty program in enterprise organizations are discussed: an institutional approach where the bug bounty program is managed in-house, and utilization of a bug bounty services provider. For both options, this work offers detailed instructions for launching, operating and managing a bug bounty program. This includes aspects such as creating the rules of engagement, assembling the operations team, defining success metrics, selecting the right targets and processing vulnerability reports.

To test the concepts of this thesis, a case study for a major project is conducted. Expert interviews are used as an additional methodology to evaluate the performance of these concepts. The results conclude that crowdsourced offensive security is an effective measure to improve IT security in enterprise organizations. The results also show that crowdsourced offensive security does not replace traditional offensive security methods. Instead, it serves as a complementary measure and increases the effectiveness of other offensive security methods when used simultaneously.

Keywords

Enterprise IT, Offensive Security, Crowdsourced Security, Bug Bounty

Contents

List of Figures

List of Tables

List of Abbreviations

A-SIT	Austrian Center for Secure Information Technology
APTs	Advanced Persistent Threats
BBP	Bug Bounty Program
CIA	Confidentiality, Integrity and Availability
CISO	Chief Information Security Officer
CMM	Capability Maturity Model
CVSS	Common Vulnerability Scoring System
DDOS	Distributed Denial of Service
ENISA	European Union Agency for Cybersecurity
GDPR	European General Data Protection Regulation
HTTPS	Secure HTTP communication
ISMS	Information Security Management System
ISO	International Organization for Standardization
KPM	Key Performance Measures
NIST	National Institute of Standards and Technology
OWASP	Open Web Application Security Project
PDCA	Plan-Do-Check-Act
PGP	Pretty Good Privacy
TCT	TIBER Cyber Team
TIBER-EU	European Framework for Threat Intelligence-based Ethical Red Teaming
TTI	Targeted Threat Intelligence
VRP	Vulnerability Reward Program

1 Introduction

More than ever, today's society relies on technology in order to function properly. Individuals, small businesses and enterprise organizations all use computer systems to complete daily activities. Sensitive information like social security numbers, credit card information and medical records are stored and accessible online. If the internet was a place to find and share static information in the past, nowadays it is the main driver of commerce and plays an important role in diplomatic relations[1].

The rise of its popularity and opportunity to generate income, inevitably also attracted criminals. The ability to remotely and anonymously access data made the internet an attractive target for financial crimes. As shown in the 2019 data breach investigations report[2], 71% of attacks were financially motivated. Other motivations such as espionage resulted in large investments for finding ways to hack into systems.

Since the introduction of computer systems and the internet, software has been plagued by vulnerabilities[1]. These vulnerabilities often have been exploited for financial gain, breaching confidentiality, integrity and availability of production systems. In order to protect themselves from attacks, companies have adopted security programs to find and patch their vulnerabilities.

1.1 Problem Statement

A fundamental part of every security program is an offensive security approach, where assets are tested for vulnerability identification, according to Proctor and Byrnes[3]. Kim[4] argues that in this approach, penetration testing and red-team exercises are used to identify weaknesses in target systems and perform simulations of real world attacks. Although these methods have proven to be successful, they lack the ability to scale with rising demand, partly due to limited professional force[2]. Bug bounties attempt to fix this gap, by offering a crowdsourced model for security assessments. By opening the target under test to numerous testers at a time and incentivising security research via cash rewards, companies can benefit from techniques and skillsets of a large crowd of testers with different backgrounds. Whereas penetration testing and red-team exercises show the security posture of an organization at a single point in time, bug bounties often run for a long (possibly infinite) time and as such, offer continuous security testing at the benefit of a pay-for-results model. The concept of crowdsourced security is not new, with a first appearance by Netscape in 1995 and formalized by Mozilla in 2004, as explained by Munaiha and Meneely[5]. Increases in rewards and emergence of bug bounty platforms have resulted in popularity of this concept among hackers and companies wanting to improve their security posture[3].

According to Maillart et al.[6], implementing a successful bug bounty program has proven to

[1] We analysed 18 years of software vulnerabilities: Here's what we found, https://techmonitor.ai/cybersecurity/software-vulnerabilities-cves-analysed, last visited on 05/24/2021
[2] The Cybersecurity Workforce Gap, https://www.csis.org/analysis/cybersecurity-workforce-gap, last visited on 05/24/2021
[3] Bug Bounty Programs are Growing Up Fast and Paying More, https://www.darkreading.com/vulnerabilities—threats/bug-bounty-programs-are-growing-up-fast-and-paying-more/d/d-id/1328428, last visited on 05/24/2021

seriously increase the number of vulnerabilities found as well as their severity, however it comes with its own challenges. Due to the nature of bug bounties, which is inviting amateurs as well as professionals to perform analysis upon the target systems, ambivalent quality of reports is an issue that must be considered in interpreting the metric of quantity. Although a recent mapping study on bug bounty research by Magazinius et al.[7] reveals that 41 out of 72 papers focus on the organizer of a bug bounty, Al-Banna et al.[8] show that a set of pre-adoption fears and post-adoption issues have prevented some organizations from using crowdsourced security. This is especially true for enterprise organizations, as shown by HackerOne's yearly report in 2019[9], which states that despite the benefits of crowdsourced vulnerability discovery, 93% of Forbes Global 2000 companies do not have a known vulnerability disclosure policy. A possible cause for the slow adoption of crowdsourced security by enterprise organizations could be the to lack of policies and integration concepts for this new offensive security measure.

1.2 Motivation

While bug bounties have been raising in popularity within the IT Security industry in recent years[4], they have not gotten the same attention from the research field, as concluded by both Fryer et al.[10] and Magazinius et al.[7]. In their literature reviews, they show the current state of the art of this field from a scientific perspective.

In the current state of the art, empirical analysis of bug bounty programs is the most prevalent methodology used. For example, Ruohonen et al.[11] analyzed vulnerabilities reported through the Open Bug Bounty platform between 2015 and 2017. Zhao et al.[12] collected publicly available data from HackerOne and Wooyun, and included other platforms such as Bugcrowd and Cobalt in a later study [13]. Other publications show different aspects of running a bug bounty, such as practices and issues of running a bug bounty program by Malladi et al.[14], and challenges of running a bug bounty program at scale by Zhao et al.[15]. The perspective of the security testers is also covered by Luna et al.[16], who investigate patterns of activity and compare productivity of security researchers in the bug bounty programs.

As mentioned in the problem statement of this thesis, adoption by enterprise organizations is still limited. This is also noted by Zhao et al.[15] who demonstrate that missing regulations by policies hinder a wider approach of bug bounties, and that more research efforts are needed in constructing such policies for vulnerability research. Therefore, this thesis is going to discuss an approach for integrating a bug bounty program into enterprise organization's offensive security efforts.

1.3 Methodological Approach

Initially, scientific literature will be used to define the foundations of IT Security, the threats of today's digital world, and why IT Security is needed. The offensive security aspects of an organization's security program are covered next, showing the different methods that can be applied to test resilience against cyberattacks. Going further, the concept of bug bounties will be covered from a research point of view. Using scientific literature research, the basic principles of how bug bounties work, why they are a promising approach to offensive security, and how organizations can profit from them will be analyzed.

[4] Bug Bounty Programs are Growing Up Fast and Paying More, https://www.darkreading.com/vulnerabilities—threats/bug-bounty-programs-are-growing-up-fast-and-paying-more/d/d-id/1328428, last visited on 05/24/2021

Building upon the knowledge of this theoretical study, this work will research an approach for integrating crowdsourced offensive security into enterprise IT. Development of an effective vulnerability disclosure policy will be researched, and an analysis of the different offerings of bug bounties platforms with regards to integration possibilities will be provided. Additionally, through analysis of different aspects required to run a bug bounty program, improving enterprise IT Security with crowdsourced offensive security will be researched. A comparison of bug bounties against other offensive security measures will be performed and argued whether and if, how bug bounties complement established offensive security techniques.

Finally, expert interviews will be conducted to determine the effectiveness of the proposed approach in enterprise environments. A predefined questionnaire will be used, and asked during conference calls where possible. The selection of IT Security experts will be focused on individuals with experience in managing bug bounty programs in enterprise organizations.

1.4 Expected Results

This thesis is expected to provide an approach for integrating bug bounty programs into enterprise IT Security. According to A. Kuehn and M. Mueller [17], case studies and reports from bug bounty platforms show promising results of using crowdsourced security to increase the number of vulnerability reports against the available attack surface. To evaluate effectiveness, an analysis of certain metrics like completeness, number of findings and severity will be provided.

To comprehend how to incorporate this emerging offensive testing method in an efficient manner, an analysis of possible methods to integrate bug bounties into enterprise IT Security policies is shown. It is expected that enterprise organizations will have trade-offs in using a bug bounty platform versus managing a bug bounty program in-house.

Finally, this thesis will show that enterprise organizations can successfully use crowdsourced offensive security to complement other offensive security efforts, and as such, improve their security posture.

The objective of this thesis is to answer the following research questions:

- How can enterprise organizations integrate a bug bounty program as an offensive security measure?
- Is crowdsourced offensive security an effective measure to improve the security posture of an enterprise organization?
- Can bug bounties complement penetration tests?

2 IT Security Basics

The term „IT Security“ represents a broad field associated with many aspects of protecting computer systems against threats. Computer systems are nowadays a part of daily life, with information being stored and constantly flowing through the internet. This brings a heap of security risks to light, which according to Easttom[1] may be leveraged for various reasons such as financial gain, political motives etc. In order to discuss the problem field of IT Security in a scientific manner, it is necessary to agree upon a formal definition of this problem space first.

2.1 Defining Information Security

Because it is such a broad term, defining IT Security is not an easy task. Many fields have their own view on what security means for their IT systems, which leads to a number of different definitions. According to the National Institute of Standards and Technology (NIST) [1], the term „IT Security“ is defined as:

> *„the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability“.*

Another definition considered in this thesis is given by Anderson[18] who defines information security as:

> *„A well-informed sense of assurance that information risks and controls are in balance.“*

Bosworth et al.[19] defines IT Security as:

> *„the state of being free from danger and not exposed from accidents or attack “.*

Further definitions by Bishop[20] and Mellado et al.[21] either use confidentiality, integrity and availability as elements to define information security, or expand on them to become more comprehensive.

For the purpose of this thesis, the given definition by NIST[1] will be used and extended to include authenticity and nonrepudiation. Each of the individual elements (confidentiality, integrity, availability, authenticity, nonrepudiation) will be discussed in more detail in section 2.1.2.

Since these elements only describe the security goals of an organization, a definition of potential threats against these goals is required and provided in section 2.1.4. The probability that such a threat occurs, and the potential damage it may cause, represent the actual risks that IT Security faces. A detailed definition of risks is given in section 2.1.5.

[1] NIST glossary, https://csrc.nist.gov/glossary/term/information_security, last visited on 05/24/2021

2.1.1 The Need for IT Security

As explained by Easttom[1], today the internet is not only used for video and social media, but it represents the foundation of economic and political relations, and is used for critical systems such as power distribution, health care, law enforcement, and national defense[2]. Many business transactions are performed online and a great deal of personal information is stored in various devices connected to the internet. The technology research firm Gartner reports 14.1 billion connected devices in 2019, and that this number will reach 25 billion by 2021[3]. Even industrial automation systems such as power generation, gas and water supply, and transportation have been pressured to connect to the internet in an attempt to make fast and cost effective decisions, according to Dzung et al.[22]. All of these devices run code written by people and are prone to errors. Some of these errors can be leveraged to make systems do actions they are not supposed to do, such as transfer funds to a malicious account or redirect mail. These errors are called vulnerabilities and will be defined in section 2.1.3.

The motivation for a malicious actor to attack these systems is best described by looking at data breaches. According to Privacy Rights Clearinghouse[4], there have been more than 4500 breaches made public since 2005, and the personal information of more than 816 million individuals have been exposed. Widup et al.[23] reports over 2100 breaches in 2015 alone, with the top three affected industries being public, information and financial services. Some examples of recent data breaches worth taking a note as given by Symantec[5] are:

- Yahoo, 2013: 3 billion user accounts
- Equifax, 2017: 145.5 million accounts
- Anthem, 2015: 78.8 million customers
- U.S. Office of Personnel Management, 2015: 21.5 million current, former, and prospective federal employees' personal information
- Friend Finder Networks, 2016: 412 million accounts, including email addresses and passwords

Data breaches not only have a negative effect on a company's stock as per Rosati et al.[24], but they also cost a lot of money. Cybersecurity ventures[6] predict that cybercrime will reach a cost of $6 trillion annually by 2021. In the yearly „Cost of a Data Breach" report by Ponemon Institute[25], it is shown that the average cost for a data breach in 2019 was $3.92 million. They also show that companies which participated in the study, estimate the probability of experiencing a data breach in the next two years at 29.6%. Due to this, the IT Security marked has experienced tremendous growth and is expected to reach 248.26 billion U.S. dollars by 2023, according to Statista[7]. Additionally, many countries have created legislation around data breaches. The European Union

[2] Internet Infrastructure, https://www.gao.gov/products/GAO-06-672, last visited on 05/24/2021

[3] Gartner Identifies Top 10 Strategic IoT Technologies and Trends, https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends, last visited on 05/24/2021

[4] Data Breaches | Privacy Rights Clearinghouse, https://privacyrights.org/data-breaches, last visited on 05/24/2021

[5] A Brief History of Data Breaches, https://www.lifelock.com/learn-data-breaches-history-of-data-breaches.html, last visited on 05/24/2021

[6] Cybercrime Damages $6 Trillion By 2021, https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016, last visited on 05/24/2021

[7] Size of the cybersecurity market worldwide, from 2017 to 2023, https://www.statista.com/statistics/595182/worldwide-security-as-a-service-market-size, last visited on 05/24/2021

passed the general data protection regulation[8] in 2016, which requires companies to notify the respective authority within 72 hours after detecting a breach. Although the US does not have federal legislation for data breaches, such regulation is being proposed and worked on[9]. Meanwhile, each of the 50 member states already have implemented their own versions of it. A comparison of these legislations is given by Hayes[26]. As is evident, while there are many benefits from the internet, it comes at a cost.

2.1.2 The Confidentiality, Integrity and Availability (CIA) Model

In section 2.1.1 it is shown that many definitions for IT Security use the elements of confidentiality, integrity and availability. These are often referred to as the CIA model (or triad), taking the first letter of each element. Together they embody the fundamental security objectives for data, information and computer services according to Stallings and Brown[27]. Figure 2.1 shows these elements and the relationship between them. As can be seen, IT Security must address all three goals to ensure protection of systems. Depending on security requirements, there can be situations where only one or a combination of two will suffice.

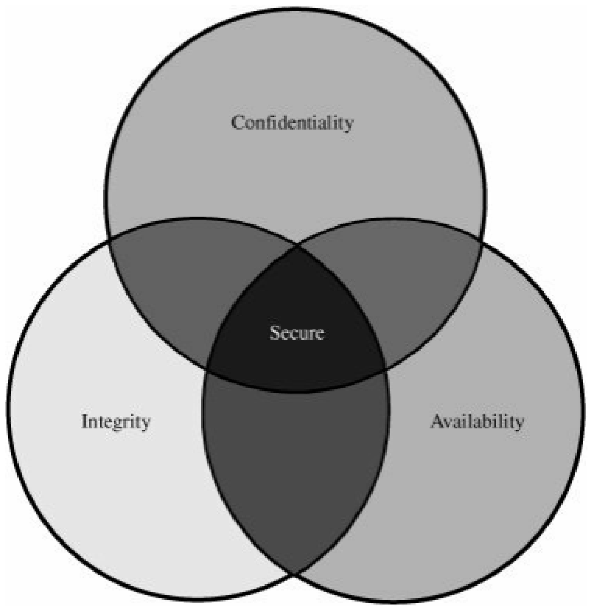


Figure 2.1: The relationship between confidentiality, integrity and availability [28]

Confidentiality Confidentiality is a concept which refers to the protection of data from unauthorized access. Bishop[20] describes it as the concealment of information or resources, and the need to keeping it secret. For example, bank account information about balance should be kept secret from anyone but the account owner, to prevent criminals from going after the person. Another example of the requirement for confidentiality is the protection of patient medical records in healthcare systems.

Bishop[20] also notes that access control mechanisms are used to support confidentiality, one of

[8] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL, https://eur-lex.europa.eu/eli/reg/2016/679/oj, last visited on 05/24/2021

[9] Latest draft of U.S. federal privacy bill sidesteps key sticking points, https://www.reuters.com/article/us-usa-congress-privacy/latest-draft-of-u-s-federal-privacy-bill-sidesteps-key-sticking-points-idUSKBN1YM2NF, last visited on 05/24/2021

which is cryptography. As Schneier[29] explains, cryptography is the science of keeping messages secure by encryption and decryption. Encryption is a process wherein a mathematical function is used upon the data to scramble and make it inaccessible without the key. The key (or password) can be used to „unscramble“ the data back into meaningful information using another mathematical function in the decryption process [29].

According to Andress[30], confidentiality can be compromised in many ways. A person looking over the shoulder while a password is being typed in, an e-mail attachment sent to the wrong person, or a malicious actor penetrating the systems of a target organization and obtaining data. Any case where access control mechanisms fail to protect the data from unauthorized use is considered a loss of the element of confidentiality.

Integrity Stallings and Brown[27] define integrity as guarding information against unauthorized modification or destruction. They further divide integrity into two subcategories: data integrity and system integrity. Data integrity assures that information is changed only by authorized personnel and within specific rules. System integrity ensures that a system is able to perform its intended functionality but cannot be manipulated to do something else[27]. Continuing with the bank account example, the services provider must ensure the integrity property in addition to confidentiality. Only authorized users, such as the bank account owner or authorized family members, should be able to modify the balance. Additionally, the balance should only be modifiable under specific rules, such as withdrawal through ATM machines or transfers via e-banking. Failure to ensure integrity could result in financial loss for either the bank itself or its clients.

According to Bosworth et al.[19], several controls can be applied to ensure integrity. Such controls can include checksums and hash totals for completeness of information, automatic checks and testing for violations of specific controls. Furthermore, Andress[30] explains that a good example of meachanisms that ensure integrity are file systems. Such systems implement permissions that control access of files, authorization of users, and commonly allow to undo changes that are undesirable[30].

Availability Pfleeger[28] defines availability as the ability to access assets at appropriate times. In other words, at any point in time, the system must be available and provide the necessary services. This applies to systems providing services as well as data whose access is required at a certain point in time. Threats to availability are commonly referred to as denial of service attacks, and will be defined in detail in section 2.1.4.

The recent rise of malicious software known as „Ransomware“ is an example of attacks against availability, as shown by Kolodenker et al.[31]. The attacks carried out by ransomware encrypt all of the victim's files using a key (or password) which is only known to the attacker. The attacker then contacts the victim and extorts money in exchange for decrypting the files. Until the money is delivered, the victim files are essentially held ransom by the attacker, hence the name „Ransomware“. The antivirus manufacturer Kaspersky provides detailed research into WannaCry[10], a ransomware attack that infected 230,000 computers globally in May 2017. Among the affected computers were thousands of NHS hospitals and surgeries across the United Kingdom, with an estimated cost of damage of £92 million. 150 different countries were affected by WannaCry, resulting in a total cost of damage estimated at $4 billion in losses across the globe. It is not uncommon for organizations to pay the ransom, as the cost of recovering from an attack is of-

[10] What is WannaCry ransomware?, https://www.kaspersky.com/resource-center/threats/ransomware-wannacry, last visited on 05/24/2021

ten exponentially higher than the requested ransom[11]. For example, the city of Riviera Beach in Florida agreed to pay $600,000 to hackers who took over their systems[12]. In another case, Indiana county paid $130,000 to regain access to their computers infected with ransomware.[13]

Authenticity As explained by Bosworth et al.[19], the integrity element of the CIA model only ensures that information is whole or complete, and has not been tampered with. However, it does not prove the validity or correctness of information. Therefore, the CIA model is extended to include authenticity. Authenticity is an element that ensures the information is what it is expected to be, and comes from a trusted source.

As an example, consider a message sent from bob to alice. While integrity ensures the message was not modified in transit, authenticity provides assurance that the message was indeed sent from bob, and not a malicious attacker pretending to be bob. As shown by Aumasson[32], the most common form of ensuring authenticity is by using digital signatures. In a digital signature, the sender (bob) signs the message using a key or password which is private and known only to the sender. The receiver (alice) can then verify the message did in fact come from bob, because it is signed using bob's key and bob is the only person that has access to it.

Nonrepudiation Nonrepudiation is closely related to authenticity. As explained by Andress[30], nonrepudiation is the existence of sufficient evidence such as an individual cannot deny a statement or action performed by them. Continuing with the previous example, whereas authenticity provides the assurance that bob is the rightful sender of the message, nonrepudiation ensures that bob cannot deny the message is from him. According to Aumasson[32], digital signatures can be used for nonrepudiation as well as authenticity. A verified signature can be used to demonstrate that a message was signed by a particular (private) key, and therefore, it can be attributed to the holder of that key.

2.1.3 Vulnerabilities

According to Bishop [20], in computer science the term „Vulnerability“ is defined as lapses in procedures, technology or management, allowing unauthorized access or actions to be performed by a malicious actor. A vulnerability is sometimes also called a „security flaw“.

There are many causes for vulnerabilities to occur, and providing a standard way to categorize them has proven difficult, as shown by Tripathi and Singh[33]. For example, the National Vulnerability Database[14] maintains a list of vulnerability categories that currently has 126 entries. The MITRE corporation[15] provides a classification by software development, hardware design, and research concepts with future sub-categories. They also maintain the common vulnerabilities and exposures list[16], which is essentially a list of all publicly known vulnerabilities that have at least one reference available.

[11] 2018-2020 Ransomware statistics and facts, https://www.comparitech.com/antivirus/ransomware-statistics, last visited on 05/24/2021

[12] Florida city pays $600,000 to hackers who seized its computer system, https://www.cbsnews.com/news/riviera-beach-florida-ransomware-attack-city-council-pays-600000-to-hackers-who-seized-its-computer-system, last visited on 05/24/2021

[13] Indiana county meets $130,000 ransomware demand, despite advice against payment, https://www.cyberscoop.com/indiana-ransomware-la-porte-county, last visited on 05/24/2021

[14] NVD CWE Slice, https://nvd.nist.gov/vuln/categories, last visited on 05/24/2021

[15] Common Weakness Enumeration, http://cwe.mitre.org, last visited on 05/24/2021

[16] Common Vulnerabilities and Exposures, http://cve.mitre.org, last visited on 05/24/2021

One of the most prevalent sources of vulnerabilities arise from programming errors, according to Kuhn et al.[34]. In their analysis of vulnerability trends from 2008-2016, they classify vulnerabilities in 3 categories: configuration, design, and implementation errors. They show that implementation errors (essentially programming errors) account for 72.9% of the vulnerabilities. They also note that, depending on the programming language used, two types of vulnerabilities account for more than a third of implementation errors. Those are basic input validation errors and buffer errors. This shows that typically, vulnerabilities arise during software development and that one good reason to keep software up to date, is to receive the latest fixes for publicly disclosed vulnerabilities.

Due to the different types of vulnerabilities that exist, and what an attacker can achieve with them, there are a couple of means to rate the severity of a vulnerability. The Common Vulnerability Scoring System (CVSS)[17] represents a published standard for rating vulnerabilities, which takes factors such as the attack vector, the privileges required to perform the attack, the impact on confidentiality, integrity and availability etc. into account. It is the most common form of determining the severity of a vulnerability. Alternative systems such as Bugcrowd's Vulnerability Rating Taxonomy[18] use a scoring system of „Critical“ to „Informational“. In all cases, the purpose is to identify the impact a vulnerability can have, and make decisions based on that. For example, an organization can decide to prioritize fixing critical vulnerabilities over low impact ones.

To take advantage of a vulnerability, an attacker usually uses an „exploit“. As Stallings and Brown[27] explain, in most cases an exploit simply represents code written specifically for a vulnerability or set of vulnerabilities, which makes use of the vulnerability to perform some unauthorized actions benefiting the attacker. However, an exploit can also be a physical attack or sequence of commands executed to leverage the vulnerability, as shown Bosworth et al.[19]. They use the broad term „Tool“ to define the wide variety of methods available to exploit vulnerabilities in computer systems. Many exploits are published online in form of scripts, and made available to the public through databases and websites like Exploit DB[19]. As Maurushat[35] explains, there has been much debate about whether public disclosure of vulnerabilities helps improve the security of software. Maurushat argues that the movement towards disclosure of vulnerabilities is a response against the insecurity of software and hardware, with the hope that it will provide incentive for faster patching. Hence the existence of such online resources like Exploit DB. Still the issue of disclosing vulnerabilities and ready-to-use exploits is a constant discussion topic within the computer security industry, with pros and cons for either side.

2.1.4 Threats

In computer security, a threat is defined by Bishop[20] as a potential violation of security. This violation does not need to occur, the mere probability that it may occur makes it a threat. An actioned threat becomes an attack, which will be further defined in section 2.1.6. Threats are also important when risk assessments are performed, which will be covered in section 2.1.5.

There can be many different threats that must be considered as part of a security program of an organization. The EU agency for cybersecurity (ENISA)[20] provides a yearly report listing top threats. In their latest report [36], they found 15 top threats based on analysis of cyberthreat re-

[17] Common Vulnerability Scoring System, https://www.first.org/cvss, last visited on 05/24/2021
[18] Bugcrowd's Vulnerability Rating Taxonomy, https://bugcrowd.com/vulnerability-rating-taxonomy, last visited on 05/24/2021
[19] Exploit Database, https://exploit-db.com, last visited on 05/24/2021
[20] European Union Agency for Cybersecurity, https://www.enisa.europa.eu, last visited on 05/24/2021

lated information found publicly. Threats that are relevant to crowdsourced offensive security are covered in the paragraphs below.

Malware is one of the most significant categories of threats in computer security, according to Stallings and Brown[27]. They define malware as a covertly inserted program with the intent to compromise confidentiality, integrity or availability of data. Another definition of malware is given by Bosworth et al.[19], who define malware simply as programs or files that are harmful to the end user. They note that the term „malware“ is used to refer to the entire category of software coded with malicious intent. An example of malware was covered in section 2.1.2 where ransomware is discussed. Unlike exploits, malware are inherently malicious and designed for activities such as damaging devices, demanding ransom or stealing sensitive data[21]. On the other hand, an exploit can be used to deliver malware, but the exploit code is not the malware itself. Although malware and exploits are used in combination, they represent two different threats which require individual defenses.

Web application attacks have been the top published vulnerabilities in the past years, and some of the most dangerous threats according to Mitropoulos[37]. These attacks involve improper validation of untrusted input, security misconfigurations, broken authentication and session management etc. The open web application security project (OWASP)[22] provides a list of top 10 vulnerabilities, and in the last three Top Ten lists (2007, 2010, 2013), injection attacks have dominated the top five positions. This matches the results of Kuhn et al.[34] that show injection attacks accounting for 72.9% of vulnerabilities discovered.

Web based attacks are considered those attacks which make use of web systems and services as the main surface for compromising a victim, according to ENISA[36]. These are different from web application attacks, which attack the web application itself. Web based attacks are considered one of the most important threats due to the wide usage of the web[23], which offers a big attack surface and pool of victims. Any user that surfs the world wide web is a potential victim, and can be exposed to web based attacks by simply visiting a malicious web page.

Phishing is the activity of sending malicious emails and can be used in two ways according to Diogenes and Ozkaya[38]: as part of information gathering, or as an active form of attack. While used in information gathering, phishing emails typically attempt to extract valuable information from the victim. When phishing is the actual attack itself, the emails usually contain attachments with malware inside them, or link to websites that contain malware. In both scenarios, emails are disguised as coming from a trusted source. The famous Sony hack[24] was accomplished using phishing, where attackers sent fake Apple ID verification emails to key executives of the company, and intercepted their passwords. Using those intercepted credentials, the attackers were able to gain access to Sony’s internal data which, among other things, included information of movies which were in the works[25].

[21] Malware vs. Exploits, https://www.paloaltonetworks.com/cyberpedia/malware-vs-exploits, last visited on 05/24/2021
[22] OWASP Top Ten, https://owasp.org/www-project-top-ten, last visited on 05/24/2021
[23] Internet usage worldwide - Statistics & Facts, https://www.statista.com/topics/1145/internet-usage-worldwide, last visited on 05/24/2021
[24] Sony Hackers Used Phishing Emails to Breach Company Networks, https://www.tripwire.com/state-of-security/latest-security-news/sony-hackers-used-phishing-emails-to-breach-company-networks, last visited on 05/24/2021
[25] ‘Doctor Who’ Set to Become Film, Reveals WikiLeaks Trove of Hacked Sony Emails, https://www.tripwire.com/state-of-security/latest-security-news/doctor-who-set-to-become-film-reveals-wikileaks-trove-of-hacked-sony-emails, last visited on 05/24/2021

Denial of service is a threat specifically aimed at violating the availability of a system, as explained by Abliz[39]. A common technique to perform a denial of service attack is making more requests than the system can handle at a single time, therefore delaying other (legitimate) requests. ENISA[36] ranks denial of service 5th in their list of top threats, and note that these threats are still on the rise. This is further confirmed by looking at the two largest denial of service attacks[26] ever registered, both of which happened as recent as 2018 and only 5 days apart each other.

A variant of denial of service is called „distributed denial of service (DDOS)“. As Easttom[1] explains, this attack is launched from many different machines at once, hence the „distributed“ in the name. These machines can be any device connected to the internet, and which has been infected with malicious software that gives the attacker control over it. According to Easstom[1], this sort of denial of service attack is easy to do, and it can be hard to stop.

2.1.5 Risks

Risks are commonly discussed in combination with a risk assessment. Stallings and Brown[27] define a risk to be the potential of loss, calculated from the likelihood of a threat being exploited, and the harmful consequence that will result to the organization. As shown, risks are closely associated with vulnerabilities and threats of them being exploited.

As Shedden et al.[40] explains, the first step to performing a successful risk assessment is the systematic identification of assets in need of protection. An asset is considered any system that is vital for the organization to perform its functions. This can be hardware as well as software, documentation and even people. Once these assets have been identified, the next step in the process is to identify the risks towards which these assets are exposed. According to Stallings and Brown[27], a risk is calculated as follows:

*Risk = (Probability that threat occurs) * (Cost to organization)*

By calculating the risk for each threat, an organization can make smart decisions and maximize benefit from resources invested in security. Each of the risks are then combined into an overall risk rating for the organization.

2.1.6 Attacks

An attack is defined as a series of steps taken by an attacker to achieve unauthorized access, as shown by Pfleeger and Pfleeger[28]. It is essentially the realization of a risk as defined in the previous section, where a malicious attacker finds a vulnerability and exploits it for their benefit. An attacker can be a human or another system, depending on the scenario.

The motivation behind an attack can be many fold. Bosworth et al.[19] categorize attacks based on their objective into attacks with political motivation, financial gain, objective of giving damage and the thrill/challenge of being able to successfully execute an attack. They also categorize attackers into Hackers, Spies, Terrorists, Corporate raiders, Professional criminals, Vandals and Voyeurs, based on the purpose of the attack. As shown by Verizon's 2019 Data Breach Investigations Report[2], professional criminals with financial motivation are the top driver for cyber attacks, followed by espionage, fun, and other motivations. It is not always easy to determine the

[26] Famous DDoS Attacks, https://www.cloudflare.com/learning/ddos/famous-ddos-attacks, last visited on 05/24/2021

motive behind an attack. Some attacks occur because the target is popular, such as the Sony breach mentioned in section 2.1.4. Other attacks simply pick easy and random targets, usually vulnerable against publicly known vulnerabilities.

As shown by Donaldson et al.[41], for an organization to protect themselves against attacks, an effective cybersecurity program is required. The purpose of such a program is to protect the organization in a cost-effective manner, by balancing available resources with the potential threats and attacks. An integral part of a cybersecurity program is the assessment of an organization's resilience against attacks, by using offensive security as a testing mechanisms. Using this, attacks can be emulated without taking the risk of consequences from an actual attack.

2.2 An Offensive Approach to Security

As mentioned in the previous sections, an organization needs to have processes in place to protect against the various threats and risks they face. In their publication about security and privacy controls for federal information systems and organizations, the National Institute of Standards and Technology provides a risk management framework[42] which details one possible implementation of a cybersecurity program. Figure 2.2 shows this process and the individual steps. In the first

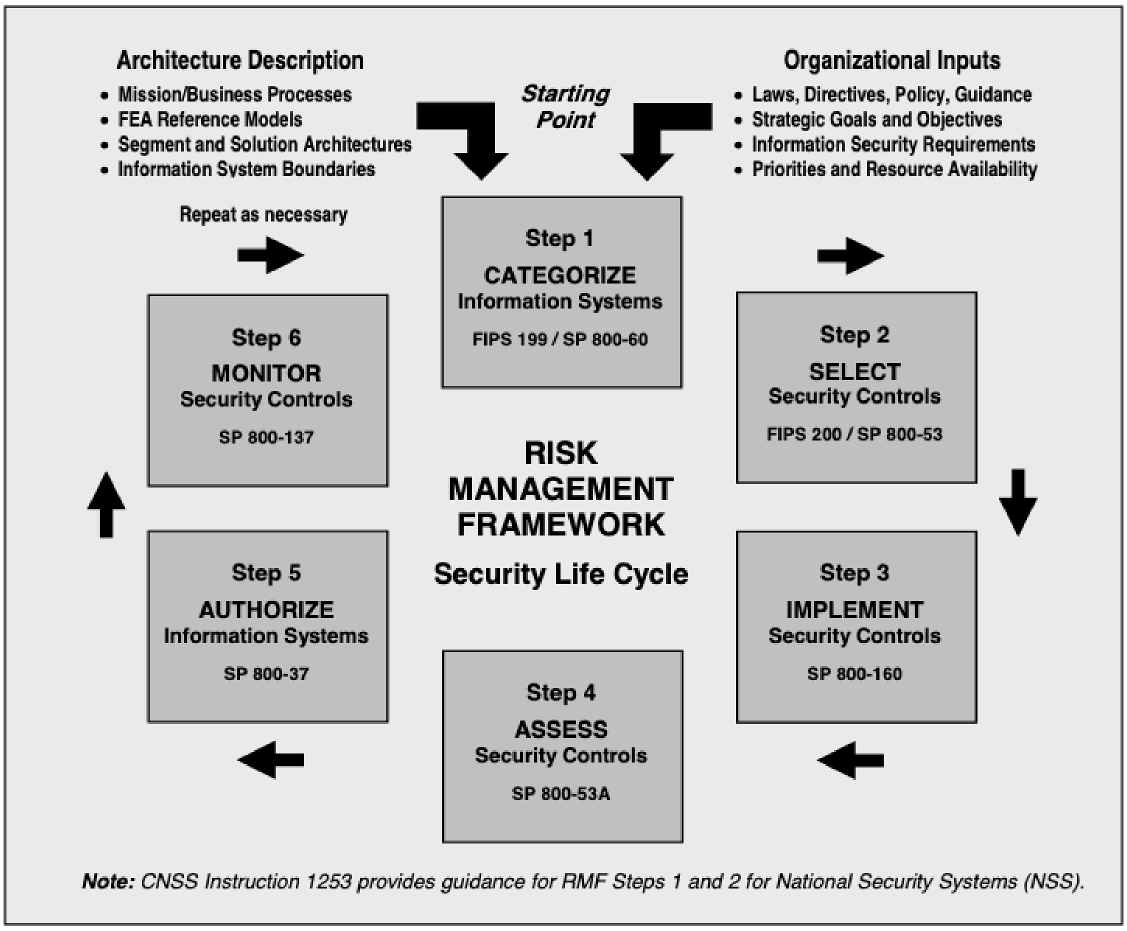

Architecture Description
• Mission/Business Processes
• FEA Reference Models
• Segment and Solution Architectures
• Information System Boundaries
Starting Point
Organizational Inputs
• Laws, Directives, Policy, Guidance
• Strategic Goals and Objectives
• Information Security Requirements
• Priorities and Resource Availability
Repeat as necessary
Step 1 CATEGORIZE Information Systems FIPS 199 / SP 800-60
Step 2 SELECT Security Controls FIPS 200 / SP 800-53
Step 3 IMPLEMENT Security Controls SP 800-160
Step 4 ASSESS Security Controls SP 800-53A
Step 5 AUTHORIZE Information Systems SP 800-37
Step 6 MONITOR Security Controls SP 800-137
RISK MANAGEMENT FRAMEWORK
Security Life Cycle
Note: CNSS Instruction 1253 provides guidance for RMF Steps 1 and 2 for National Security Systems (NSS).

Figure 2.2: Risk management system by NIST[42]

step, the identified assets as described in section 2.1.5 are categorized based on importance. Step 2 and 3 select and apply protections, and step 4 is where these protections are tested using offensive security mechanisms. In step 5, systems are authorized to operate based on risks and results of implemented security measures, and step 6 is an ongoing monitoring of the effectiveness of these measures. In the next sections, we will dive into the details of using offensive security as a means to increase the security level of an organization. As shown by Arce and McGraw[43], computer security is becoming increasingly sophisticated, and understanding the nature of attacking techniques is an important step towards building more secure systems. This increased complexity in computer systems has pushed the boundaries of IT Security, therefore, a thorough analysis and testing of systems has become part of every security design and implementation. Designers need to understand the motivation and techniques behind attackers, in order to effectively build protections against them.[43]

2.2.1 Offensive Security Testing Techniques

According to Donaldson et al.[41], a cybersecurity program needs to be constantly assessed and improved to provide assurance of its capabilities, and to keep up with emerging threats. Such an assessment also helps in making business decisions by providing quantitative measurement. Donaldson et al.[41] include three elements in a cybersecurity program assessment: risk mitigation, security capabilities, and security operations. Threat scenarios are then considered for each of these elements, and analyzed from the attacker's perspective using penetration testing and red team exercises. Improvements in any of the mentioned elements will improve the entire IT Security posture of an organization, according to Donaldson et al.[41].

In their risk management framework, NIST[42] includes step 4 to assess the security protections implemented. This activity's purpose is to provide assurance to an organization about their security functionality, and ability to deliver the required security capabilities. The strength of such security capabilities is defined by the security evidence that can be derived from security policies, documentation, threat scenarios and security testing. Security testing as a producer of security evidence includes security assessments performed by the organization itself, vulnerability scanning, penetration testing and red team exercises.

NIST provides a security control called „Penetration Testing“ as part of offensive security testing[42], although in itself, this control includes penetration testing and red team exercises. The purpose of this control is to mimic activities of attackers and provide a detailed analysis of security deficiencies which must be improved.

As stated above, penetration testing and red team exercises are common techniques for offensive security testing. Utilization of such offensive security techniques is often required to be compliant with regulation such as PCI DSS[44], HIPAA and ISO 27001[45].

Penetration testing is a methodical and rigorous testing technique, according to Kim[4]. It is used to scan networks and systems for vulnerabilities, take advantage by exploiting them and document the process. According to Bosworth et al.[19], penetration tests commonly follow a defined methodology and are limited in time and scope. The result of a penetration test is a well documented report, which includes actionable results. Usually, such results include patching of vulnerable systems and adding or improving additional defenses.

Oakley[46] argues that the most accurate description of a **red team** is *adversary emulation*. Through such an emulation exercise, an organization is able to understand and improve its defense, detection, and resilience capabilities against real threat actors. According to Kim[4], red-team exercises are not as methodical as penetration tests, and the techniques used depend on the objective of the campaign. Kim[4] argues that a common objective of a red-team exercise is evaluating the detection capabilities of a target organization. The evaluated metrics are „Time To Detect“ and „Time To Mitigate“, which represent the time from the initial occurrence of the attack, to the time it is detected by tools or personnel and responded to. Other common objectives of red-teams include infiltration, infection, exfiltration, persistence etc. Depending on the objective, the duration of the red-team exercise can be from weeks to months long, and no prior announcement to security teams is given. According to Oakley[46], these characteristics give red-team exercises advantages over other methods of security testing, because they are tailored to, and provide a unique assessment of the organizations ability to withstand complex attacks.

In the following sections, we will dive into the details of these offensive security testing techniques, understanding the approach and the exact steps taken to achieve the desired results.

2.2.2 Identifying Vulnerabilities via Penetration Testing

As has been established in the previous section, penetration tests are a methodical approach to vulnerability discovery. Weidman[47] explains that commonly, penetration testing includes a number of stages performed by the tester:

1. Pre-engagement
2. Information gathering
3. Threat modeling
4. Vulnerability analysis
5. Exploitation
6. Post-exploitation
7. Reporting

However, the engagement may differ depending on the testing perspective, according to Bosworth et al.[19]. For example, an external assessment will attempt to identify vulnerabilities originating from the internet, whereas an internal assessment assumes the role of an attacker who has already gained access to the internal network.

Oriyano[48] explains that, typically, a penetration test takes the form of a black-box, gray-box or white-box approach. In a black-box approach, very limited to no information is provided to the penetration tester. It is the most realistic form of attack, which is carried out from outside the organization. In a gray-box approach, some information is passed to the penetration tester to see what can be accomplished. Information can also be provided to ensure the desired coverage is achieved, and knowing when to stop testing. Finally, in white-box penetration testing, all available information is given to the tester in advance. This form allows for detailed analysis and in-depth testing of target systems, and is much more cost-effective compared to the other two forms of testing[48].

Pre-engagement In this phase the penetration tester and the client meet about the penetration testing engagement, according to Weidman[47]. The goal of this phase is to understand the requirements and goals of the penetration test, and ensure both sides agree upon the outcome. Important topics such as scope of the engagement, the testing window and contact information are discussed in this meeting. The scope of the engagement defines exactly which systems should be tested, and to what extend, according to Bosworth et al.[19]. This can include test systems as well as production systems. It is important to define the scope correctly, to ensure that the client is receiving the desired level of testing, and the penetration tester does not go out of bounds. The testing window defines the starting time and duration of the penetration test. This allows the IT and security departments to distinguish security alerts generated by a penetration test versus an actual security incident. Other discussion points include non-disclosure agreements from both sides, payment terms etc., as shown by Weidman[47].

Information gathering As Oriyano[48] explains, during information gathering, the penetration tester enumerates any available information about the in-scope targets. This information can come from sources such as search engines, websites, job sites and social engineering (if agreed so in the scope). Depending on the scope of the engagement, this may include information about employees, domains and email addresses, ip address ranges and more. According to Weidman[47], scanning is also part of information gathering. During scanning, the penetration tester starts to investigate the open and accessible ports of the target systems. Any open port is a potential entry point for attacking the system and services it supports, and therefore, must be carefully analyzed by the penetration tester.

Threat modeling After the information gathering phase is finished, the penetration tester starts to analyze the available information. As shown by Weidman[47], using threat modeling the penetration tester paints a picture of the security boundaries of the organization and starts thinking like an attacker. An important aspect of this phase is to recognize the organization's valuable assets and start developing a strategy for going after them.

Vulnerability analysis With a strategy in place, the penetration tester starts to actively search for vulnerabilities. This includes running automated tools to identify known vulnerabilities as well as careful studying and detection of new ones, as explained by Oriyano[48]. Vulnerability scanners are a common tool for automated vulnerability discovery. As shown by Bosworth et al.[19], these tools hold a database of known vulnerabilities, and use it to compare against information obtained from actively scanning targets. If a match is found, the tool reports that a potential vulnerability has been identified. Although powerful, a vulnerability scanner cannot compensate the critical thinking of a human according to Weidman[47]. Hence, penetration testers also perform manual analysis of targets. As shown by Bosworth et al.[19], some of the most common penetration testing techniques for manual analysis are unexpected input attacks, overflow attacks, filesystem exploits etc.

Exploitation In the exploitation phase, vulnerabilities that have been discovered are leveraged with the intention of compromising the system, as explained by Oriyano[48]. Penetration testers attempt to exploit all vulnerabilities discovered in the previous phases, although this may not always be possible depending on the size of the target. Some vulnerabilities may not be exploited at all or under specific conditions. In order to prove exploitability, Oriyano[48] lists some of the types of attacks that may appear during this phase: Password cracking, Traffic Sniffing, Session hijacking, Brute-force attacks, Man-in-the-middle attacks. However, sometimes penetration testers are required to manually exploit a vulnerability, as shown by Weidman[47]. One tool that is commonly used to efficiently create exploits is Metasploit[27].

Post-exploitation There are a number of activities in this phase. According to Weidman[47], during post exploitation information about the attacked system is gathered. This includes inter-

[27] Metasploit The world's most used penetration testing framework, https://www.metasploit.com/, last visited on 05/24/2021

esting files, dumping passwords and continuing exploitation of the target to elevate privileges. The compromised system can also be used as a pivot point to attack other systems which are not reachable, provided the scope of the engagement allows such actions. Weidman[47] also argues that this phase is used to demonstrate the damage an actual attacker can do when compromising a system. According to Oriyano[48], another activity in this phase is ensuring access is maintained. This is under the assumption that the penetration tester will come back later to perform additional tasks, or uses the system to pivot further attacks.

Reporting This is the final phase of a penetration test, where the penetration tester conveys the discovered findings in a meaningful way to the client, as explained by Weidman[47]. Creating a good report is a difficult task, as the audience is technical as well as non-technical people. Oriyano[48] suggests that a penetration testing report should start with an overview of the penetration testing process without much technical details. This section is intended for the executive audience. Further, the report should include all successful system compromises, a detailed list of all information gathered during the engagement, a detailed list of all vulnerabilities found and their descriptions, and a recommendation summary for resolving the vulnerabilities. Penetration testing reports are generally free form, however they may have a specific format when conducted as part of compliance testing[48].

Donaldson et al.[41] suggest that internet facing networks and user networks should be assessed using penetration testing at least annually. This is also the case with compliance requirements such as PCI DSS[28], and standards such as the ISO 27001 information security management standard[49]. For internal networks and critical systems, Pompon[50] suggests monthly and quarterly assessments depending on the available capacities. Ultimately, unless driven by compliance, the frequency of penetration testing and offensive security testing in typically driven by the company security policy.

2.2.3 Emulating Real World Attacks via Red Teaming

As shown in section 2.2.1, the purpose of red teaming is to emulate real world attacks and test the organization's resilience against them. Kim[4] explains that in red team campaigns, the assessment starts with a few key objectives. Such objectives are defined in the scoping phase and, according to Oakley[46], this is the most important aspect and will drive the activities of the engagement.

Oakley[46] explains that an organization's needs and goals must be defined first, to allow tailoring the red team engagement towards them. These goals are identified in discussions between the customer and the red team provider. Oakley[46] emphasises that both sides require to have technical and nontechnical personnel in these discussions, to agree on details. For example, the customer technical personnel may know about new systems that testing should be focused on, or less important systems that will be decommissioned. Nontechnical personnel from the customer will help drive the possible cost benefit and determine to what extend resources can be offered. From the red team provider, technical personnel will be important to make sure the engagement stays within the boundaries of whats possible, and nontechnical personnel helps drive the window of testing and other items such as the contract.

Another important aspect of red team exercises are the rules of engagements. According to Rehberger[51], these are a clear set of rules established and approved by leadership and legal

[28] PCI Security Standards Council Document Library, https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss, last visited on 05/24/2021

departments, to ensure permission is given for simulating real world attacks. They dictate how the assessment upon the previously agreed scope will be performed, and are typically created in the form of a rules of engagement document. Oakley[46] argues that a rules of engagement document serves three main purposes: it creates a legal foundation for the engagement, it establishes the required approval in form of signatures from both parties, and lastly it defines confidentiality and liability agreements. A rules of engagement document also specifies the activity types that are allowed (physical intrusions, social engineering, pivoting)[46], and whether testing should be performed on production versus non-production systems[51].

Due to the nature of emulating real world adversaries, a red team exercise can take many forms. Kim[4] argues that one assessment which is highly beneficial to an organization is called an „Assumed Breach Exercise“. In this type of engagement, the assumption is made that an attacker has already gained initial foothold into the internal network, in form of exploiting a vulnerability or simply a disgruntled employee turned into an attacker. The red team will try to compromise further systems and move within the internal network of the organization, emulating the attacker. The goal of this type of engagement is to test the internal security defenses and time it takes to detect an internal breach.

Another common scenario emulated by red teams are so called „Advanced Persistent Threats“ (APTs). According to Oakley[46], APTs are defined as

> *„well-resourced malicious actors in cyberspace with specific goals and organized efforts to attain them.“*

This is a challenging task for a red team, since APTs usually have more money, more resources and more capabilities than a red team can emulate. As Rehberger[51] explains, red teams typically research and learn past campaigns of an APT with the goal of simulating such events against the target organization.

Similar to penetration testing, the final phase of a red team engagement is reporting. Oakley[46] argues that even for technical people, the technical details of the red team attacks will be viewed in a different light. Therefore, emphasis is put into the less technical audience and the report is tailored that way. According to Oakley[46], a red team report should have at least the following elements: who performed the assessment, duration, rules of engagement, high-level summary of the assessment activity, detailed findings.

2.2.4 Comparison and Discussion of Offensive Security Testing Techniques

When comparing offensive security techniques, they may seem similar due to the same goal: identifying gaps in IT Security. However, despite similarities, these techniques have their own approaches as established in the previous sections.

Kim[4] provides a comparison of penetration testing and red teaming in table 2.1, showing the differences of these techniques. As shown, they share similarities such as intelligence gathering and reporting, but the overall approach is different. Due to the objective of emulating real world attacks, red team engagements require much more effort and typically last longer than penetration testing. They typically also require staffing of professionals with specialized skills, depending on the target system. Oakley[46] argues that this is particularly important for activities such as physical and wireless assessments, where experienced red teams are required to perform thorough testing. Although Oakley's comparison in table 2.1 argues that there are no rules for red team engagements, in practice, it is common to have some rules defined. As shown by Rehberger[51], red team engagements in fact have rules agreed upon by leadership and legal, they just tend to be less restrictive than rules set for penetration testing.

Another important distinction is that penetration tests are generally announced, whereas red team engagements are communicated only to key staff of organizations. Although real world attacks do not come with announcements, Oakley[46] argues that it is important to keep organizations informed about red team activity, allowing them to differentiate the assessment from a real incident.

Penetration Tests	**Red Teams**
Methodical Security Assessments: • Pre-engagement Interactions • Intelligence Gathering • Vulnerability Analysis • Exploitation • Post Exploitation • Reporting	Flexible Security Assessments: • Intelligence Gathering • Initial Foothold • Persistence/Local Privilege Escalation • Local/Network Enumeration • Lateral Movement • Data Identification/Exfiltration • Domain Privilege Escalation/Dumping Hashes • Reporting
Scope: • Restrictive Scope • 1-2 Week Engagement • Generally Announced • Identify vulnerabilities	Scope: • No Rules* • 1 Week – 6 Month Engagement • No announcement • Test Blue teams on program, policies, tools, and skills *Can't be illegal.

Table 2.1: Comparison of penetration testing and red teams[4]

3 Introduction and Discussion of Standards and Concepts for IT Security Policies

IT Security, like any other important field, has seen various development over the years. As shown by Susanto et al.[52], a number of private and government organizations have developed standards and legal regulation for information security, to ensure an appropriate level of protection. In this chapter, a brief introduction and overview of two standards from the ISO 27000 family are provided in section 3.3. According to Susanto et al.[52], the ISO 27000 family of standards is one of the most widely accepted international initiatives for the development and operation of cybersecurity programs.

However, standards themselves do not provide a framework with detailed description of the processes required to operate a cybersecurity program, as concluded by Haufe et al.[53] in their mapping study. Hence, the development and existence of such frameworks have come to light. In this thesis, the NIST[42] special publication „*Security and Privacy Controls for Federal Information Systems and Organizations*“ framework will be covered in section 3.4, followed by the „*European Framework for Threat Intelligence-based Ethical Red Teaming*“(TIBER-EU)[54]. Finally, the „*Austrian Information Security Manual*“[55] provides a perspective of the local situation in terms of standards and frameworks.

3.1 Designing a Defense Strategy

As shown by Tounsi and Rais[56], today's cybersecurity attacks are increasingly more sophisticated. With well funded and organized threat actors, IT Security must adapt and defend appropriately. Diogenes and Ozkaya[38] argue that while hardening systems and installing more security tools has worked in the past, organizations now need careful planning and a defense strategy to guide their protection efforts.

According to Diogenes and Ozkaya[38], a cyberstrategy is a documented approach developed to address the cybersecurity needs of an organization. Such a strategy brings better organization through centralized control, covers all possible attack landscapes, provides high-level tactics for ensuring security, and simplifies cybersecurity for key stakeholders. The importance of having such a strategy is shown by the various national cybersecurity strategies developed by many countries[1]. For example, austria's national cybersecurity strategy[2] defines 5 fields of actions and measures:

1. Structures and processes

[1] European National Cybersecurity Strategy Map, https://www.enisa.europa.eu/ncss-map, last visited on 05/24/2021
[2] Austrian National Cyber Security Strategy, https://www.enisa.europa.eu/topics/national-cyber-security-strategies/ncss-map/national-cyber-security-strategies-interactive-map/strategies/austrian-cyber-security-strategy, last visited on 05/24/2021

2. Governance
3. Cooperation between the government, economy and society.
4. Protection of critical infrastructures
5. Awareness raising and training

The ultimate goal of a nation's cybersecurity strategy is to improve the security and resilience of critical IT infrastructure.

A security strategy is typically documented using a number of documents. Santos[57] describes the relationship of such documents as the „policy hierarchy“, and includes the following items: policies, standards, guidelines, procedures, and baselines. A similar definition is given by Donaldson et al.[41], who uses a pyramid to depict the relationship as shown in figure 3.1.

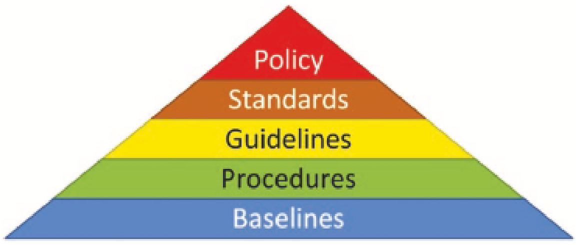


Figure 3.1: Security documentation pyramid[41]

According to Donaldson et al.[41], a policy is a high-level statement and course of action, whereas standards are documents specifying behaviour of processes, configurations and technologies to be used. Santos[57] defines guidelines as recommendations and advice to users, with the objective to help conform to a standard. A procedure is a document that has procedural steps showing how something must be done, as shown by Diogenes and Ozakaya[38]. Finally, baselines represent the application of a standard to a specific category or grouping, and according to Santos[57], they have the primary objective of uniformity and consistency.

3.1.1 IT Security Policies

Information security policies are a necessary foundation for organizational security programs, according to Knapp et al.[58]. They argue that information security policies address the confidentiality, integrity and availability elements of information systems, and represent the precondition for implementing effective protections. According to Knapp et al.[58], policies act as clear statements of management and intent. Without an approved policy document, the necessary guidance is missing and ambiguous situations lead to incorrect decisions or managerial involvement[58]. A similar definition is given by Santos[57], who argues that the role of policy is to codify guiding principles and provide those to staff that are tasked with making present and future decisions. An IT Security policy is defined as a directive which specifies how the organization is going to protect its assets and systems, as well as ensuring compliance with legal and regulation requirements. The objective of such a policy is to protect the organization, its employees, customers and partners from potential threats.[57]

According to Donaldson et al.[41], IT Security policies typically include the following elements: purpose, scope and applicability, policy statement, compliance, responsibilities. They also argue that a policy should be unambiguous, well organized, well maintained and balanced between security and business needs. Every organization is different and must continue to evolve their cybersecurity program and policies to meet the challenges of IT Security.[41].

Purpose The purpose, or policy goal, is a component that conveys the intent of the policy, as explained by Santos[57]. This can include one or multiple objectives. The purpose of a policy also specifies the security requirements and roles necessary to achieve the policy statement, according to Donaldson et al.[41].

Scope and Applicability As shown by Donaldson et al.[41], this part of an IT Security policy defines the group of people within an organization, to whom the policy applies. For example, a policy could apply to all employees, contractors, or only to a specific company branch.

Policy statement Santos[57] defines the policy statement as a high-level directive, where the rules to be followed are laid out. The policy statement provides actionable items as well as directives for specific scenarios, deviations and exceptions.

Compliance In the compliance component, penalties and disciplinary actions for violation of the policy are defined, as shown by Donaldson et al.[41]. The lack of complying with a policy could also have legal ramifications, and enforcement of the policy should be periodically assessed by IT Security[41].

Responsibilities According to Donaldson et al.[41], roles and responsibilities required to achieve the desired security goals are defined in this component. This includes responsibilities such as providing the necessary resources to achieve the policy statement, review and adaption of the policy etc.

Typically, the development of an IT Security policy is performed in a number of steps. Figure 3.2 shows the typical lifecycle of a policy development, according to Santos[57]. He further ar-

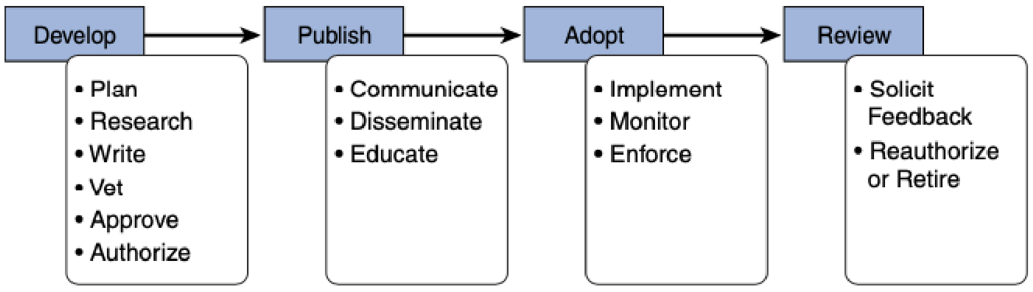


Figure 3.2: IT Security policy lifecycle[57]

gues that the success of a policy is dependent upon the development approach of an organization, and that a structured approach with defined responsibilities has a greater chance of success.

A policy can be organizational-wide and apply to the whole organization, or it can be specific to systems and functions. Donaldson et al.[41] divide policy development efforts by functional areas. As such, they provide policy samples for areas such as system administration, network security, application security, incident response and more. Easttom[1] suggests that a good set of policies for the end user include at least the following: Password policy, internet use policy, email policy, software installation policy, instant messaging policy, desktop configuration policy and personal devices policy.

As stated above, there are various policies that regulate different aspects of the cybersecurity program of an organization.

3.1.2 IT Security Controls

Stallings and Brown[27] define IT Security controls as follows:

> *„An action, device, procedure, or other measure that reduces risk by eliminating or preventing a security violation, by minimizing the harm it can cause, or by discovering and reporting it to enable corrective action.“.*

Another definition is given by Donaldson et al.[41], who use the following terminology:

> *„A security control consists of security capabilities or audit activities that are applied to an IT system or business process to prevent, detect, or investigate specific activities that are undesirable; an incident response is issued when those activities occur“.*

As is evident, in both definitions IT Security controls are considered as some form of preventive measure that reduces risk. Hence, for the purpose of this thesis, a simpler definition for IT Security controls will be used. In this work, any action or measure taken to reduce the risk of threats will be considered an IT Security control.

According to Kohnke et al.[59], there are three types of IT Security controls: preventative, detective and reactive. Preventive controls will attempt to stop an attack from happening, as the name implies. Through detective controls, intrusions can be identified and alerted upon. Finally, reactive controls attempt to apply corrective measures to situations that may arise from an attack[59]. Stallings and Brown[27] use a different classification of security controls. They categorize security controls in management controls, operational controls and technical controls. Each of these categories then includes sub-categories which are similar to Kohnke's classification. Stallings and Brown[27] argue that management controls are focused on policies, standards and other high-level planning which influences the two other categories. Operational controls are used to ensure correct implementation of the security policies and standards defined earlier, while technical controls invoke the necessary hardware and software to achieve the required security capabilities.[27]

A number of international standards provide lists of suggested security controls. For example, NIST's special publication[42] on security and privacy controls offers a classification of IT Security controls based on families. These families include controls such as awareness and training, configuration management, personnel security, physical security etc. A detailed description of NIST's special publication[42] and the security controls it offers are given in section 3.4. Similarly, the ISO's information security management standard[49] provides categories of controls such as security policies, asset management, operations security, communications security etc. For these, a detailed description is covered in section 3.3

According to Yevseyeva et al.[60], the selection of IT Security controls depends upon a number of factors. The primary objective is the assessment of security vulnerabilities, and mitigation of identified threats and risks faced by the organization. However, other factors come in to play, such as budget limitations and maintaining a balance between productivity and security. NIST[42] and ISO[49] offer selection processes as part of their publications to help with this problem. Additionally, ongoing research in this field provides novel approaches to this problem. For example, Almeida and Respicio[61] provide a framework for decision making based on investment optimizations and minimizing expected losses. Breier and Hudec[62] propose a method based on grey relational analysis combined with the TOPSIS decision making method, providing a quantitative technique for selection and prioritization of security controls. Ultimately, the decision to select appropriate security controls is left to the management of the organization, as shown by Stallings and Brown[27].

3.1.3 Creating a Cybersecurity Program

According to Schreider[63], a successful cybersecurity program is created through careful design and proper structure. If not properly designed, a cybersecurity program will have conflicting opinions of stakeholders, who use strategies that affect the completeness of the program. One of the most common ways to design a cybersecurity program is to use one of the popular standards such as NIST's special publication[42] or the ISO 27001 standard[49].

Donaldson et al.[41] argue that the following elements are needed for an effective cybersecurity program: policy, people, budget, technology, strategy, engineering, operations and assessment. Further, they explain that the chief information security officer (CISO) is the ultimate authority for cybersecurity, and directs cybersecurity policy as well as implementation. Similarly, in Schreider[63]'s definition of a cybersecurity program, everything starts from the CISO of the organization, where policies and strategic planning are done. In both approaches, policies are used to derive the responsibilities of personnel and technology. In turn, these elements ensure execution of the various tasks needed in security engineering and security operations. For the purpose of this thesis, a cybersecurity program will be defined using four components: people, policy, engineering, and operations.

People According to Donaldson et al.[41], the CISO organizes cybersecurity people and teams, and defines their roles and responsibilities. The CISO also makes a case for budget required to fund the cybersecurity program. Furthermore, Schreider[63] explains that the CISO is also responsible for creating the policies, which represent the foundation of the cybersecurity program.

Policy Donaldson et al.[41] argue that any activity performed in staffing, engineering, operations and technology must be traced back to a policy. This traceability marks the foundation of the entire cybersecurity program. Policies usually include standards, guidelines, procedures and baselines for the entire organization[41].

Engineering According to Schreider[63], engineering involves the architecture and design of secure operating environments. This includes domains such as cryptography, network security, identity and access management etc. Engineering is typically staffed with architecture and design experts, as the operational responsibility is turned over to operations.

Operations Donaldson et al.[41] argue that if cybersecurity is not maintained, it will become ineffective over time. This is where operations come into play. According to Donaldson et al.[41], security technologies must be operated and maintained to stay effective. Schreider[63] explains that this component includes a security operations center, security automation and aspects of a service desk dedicated to security.

Typically, a cybersecurity program is an ongoing process. According to Schreider[63], key performance measures (KPM) are the second most effective way to direct a cybersecurity program (policies being the first). KPMs provide a concrete way of measuring a cybersecurity program, producing metrics that show trends in positive or negative directions. Donaldson et al.[41] argue that an important aspect of cybersecurity program measurement, is the expression of metrics in everyday terms that are familiar to the organization. This allows comprehension by non-technical stakeholders as well as judgement by experts.

Through measurement of a cybersecurity program, improvements can be applied. As shown by Donaldson et al.[41], security improvements generally fall into three categories:

- **Risk mitigations** focus on the detection and prevention of known threats and attacks.
- **Security capabilities** focus on addressing unknown threats and attackers who use novel techniques.
- **Security operations** which help risk mitigation and security capabilities work together to defend against attacks on an ongoing basis.

3.2 IT Security Standards

According to Bosworth et al.[19], standards are established for providing uniformity and essential characteristics amongst different entities. They allow the understanding and comparison of such characteristics by different parties. In terms of IT Security, Moschovitis[64] explains that standards are the ecosystem of a policy, and allow the meaningful implementation of policies.

There are a number of sources for standards, as shown by Bosworth et al.[19]. Standards can be created by governments, international organizations, military, or consulting firms. For example, one national body that issues standards for federal systems in the united states, is the US NIST[3] . These standards set the minimum requirements for federal information processing systems, and other attributes such as security, privacy and best practices. Other examples of national bodies include the british standards institute[4], the german federal office for information security[5] and the european committee for standardization[6].

One of the most popular sources for standards is the international organization for standardization (ISO)[7]. They provide standards for quality management, health and safety, and IT Security standards such as the 27000 family[8]. Another popular organization is the PCI security standards council[9], which provides a number of standards, frameworks and specifications focused on payment card industry. Finally, there are also technology specific sources of standards, such as the web application security consortium[10], cloud security alliance[11], ISACA[12] etc.

As shown by Bosworth et al.[19], there are many types of standards that apply to specific industries. For example, capability standards such as the capability maturity model (CMM)[13] and

[3] NIST, https://www.nist.gov, last visited on 05/24/2021
[4] British Standards Institute, https://www.bsigroup.com, last visited on 05/24/2021
[5] Federal Office for Information Security, https://www.bsi.bund.de/DE/Home/home_node.html, last visited on 05/24/2021
[6] European Committee for Standardization, https://www.cen.eu, last visited on 05/24/2021
[7] International Organization for Standardizaiton, https://www.iso.org, last visited on 05/24/2021
[8] Family of IT Security Standards, https://www.iso.org/search.html?q=27000, last visited on 05/24/2021
[9] PCI Security Standards Council, https://www.pcisecuritystandards.org, last visited on 05/24/2021
[10] Web Application Security Consortium, https://www.webappsec.org, last visited on 05/24/2021
[11] Cloud Security Alliance, https://cloudsecurityalliance.org, last visited on 05/24/2021
[12] ISACA, www.isaca.org, last visited on 05/24/2021
[13] CMMI Institute, https://cmmiinstitute.com/resource-files/public/take-organizational-performance-to-the-next-level, last visited on 05/24/2021

ISO 9000[14] measure the organization's competence in building security-related products. Some standards specify security functionality of products, such as IETF's IPSec standard[15]. In the following sections, we will look at the details of a number of international security standards, as well as the regional and local situation.

3.3 ISO 27001 & 29147

In the following section, two popular standards created by the international organization for standardization[16] are covered: ISO 27001 and ISO 29147. The ISO 27001 standard[49] concerns itself with IT Security and management, whereas ISO 29147[65] gives guidelines for vulnerability disclosure.

3.3.1 ISO 27001

The ISO 27001[49] standard has been established for creating, implementing, maintaining and continually improving an information security management system. According to Eloff and Eloff [66], an information security management system (ISMS) addresses all aspects of creating and maintaining a secure information environment within an organization. The goal of an information security management system is to preserve the confidentiality, integrity, and availability of information systems.

As shown by Humphreys[67], the ISO 27001 standard provides a number of security processes based on a Plan-Do-Check-Act (PDCA) model. Further, Humphreys[67] argues that ISO 27001 is a risk-based standard and requires organizations to have a risk management process. Figure 3.3 shows the risk management process applied in the ISO 27001 Plan-Do-Check-Act model.

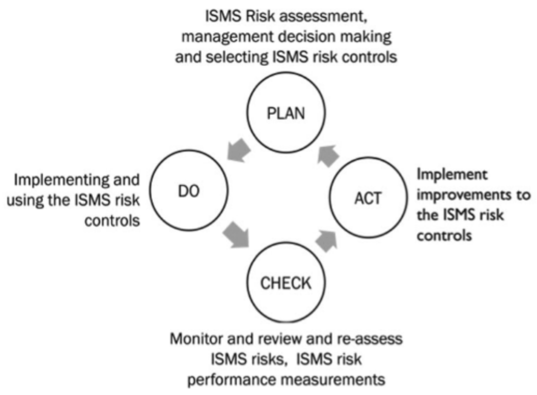

ISMS Risk assessment, management decision making and selecting ISMS risk controls
PLAN
Implementing and using the ISMS risk controls
DO
ACT
Implement improvements to the ISMS risk controls
CHECK
Monitor and review and re-assess ISMS risks, ISMS risk performance measurements

Figure 3.3: ISO 27001 risk management process according to Humphreys[67]

The ISO 27001 standard[49] has the following requirements: context of the organization, leadership, planning, support, operation, performance evaluation and improvement. Additionally, a reference for security controls and objectives is provided.

Context of the organization This requirement demands that organizations determine internal

[14] ISO 9000 Family, https://www.iso.org/iso-9001-quality-management.html, last visited on 05/24/2021
[15] IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap, https://tools.ietf.org/html/rfc6071, last visited on 05/24/2021
[16] International Organization for Standardizaiton, https://www.iso.org, last visited on 05/24/2021

and external issues that may affect the information security management system. It also expects that organizations determine interested third parties and the scope of their information security management system[49].

Leadership The leadership requirement calls for demonstrated commitment by management to ensure information security objectives are established, and that a strategy is in place to achieve them. This includes creating an information security policy that is appropriate for the organization and its aforementioned security objectives[49].

Planning In this requirement, it is expected that plans for the various processes are created. This includes general planning for treating issues identified, and the creation of processes for risk management and risk treatment[49].

Support The support requirement pushes organizations to provide the necessary resources for the implementation and maintenance of their information security management system. It includes creating and updating documentation, and gives directives on communication such as what to communicate, when to do it, with whom etc[49].

Operation This requirement expects organizations to implement and control the processes needed to meet the information security objectives defined earlier. The expected interval for performing risk assessments and treatments, as well as planned changes are determined in this requirement[49].

Performance Evaluation In this requirement, monitoring, measurement and evaluation are performed, to determine the effectiveness of the information security management system. This includes determining what to monitor and the method for doing so[49].

Improvement The final requirement expects the organization to continue improving the effectiveness of their information security management system, and to react to non-conformity with appropriate actions[49].

The enumeration of security control families as provided by ISO 27001 are shown in table 3.1. It is important that the selected controls align with the risk assessment and risk treatment plan that has been previously identified[49].

Control Families			
1.	Information security policies	8.	Operations security
2.	Organization of information security	9.	Communications security
3.	Human resource security	10.	System acquisition, development and maintenance
4.	Asset management	11.	Supplier relationships
5.	Access control	12.	Information security incident management
6.	Cryptography	13.	Information security aspects of business continuity management
7.	Physical and environmental security	14.	Compliance

Table 3.1: ISO 27001 security control families[49]

3.3.2 ISO 29147

The international organization for standardization provides guidelines for the disclosure of potential vulnerabilities in the ISO 29147 standard[65]. This includes guidelines for how to receive vulnerability information, how to publish resolution of vulnerabilities, what kind of information should be obtained, and some examples.

According to the ISO 29147 standard[65], the vulnerability disclosure processes is composed of five high-level steps: receipt of vulnerability report, verification, resolution development, release and post release. A summary of the process is shown in figure 3.4.

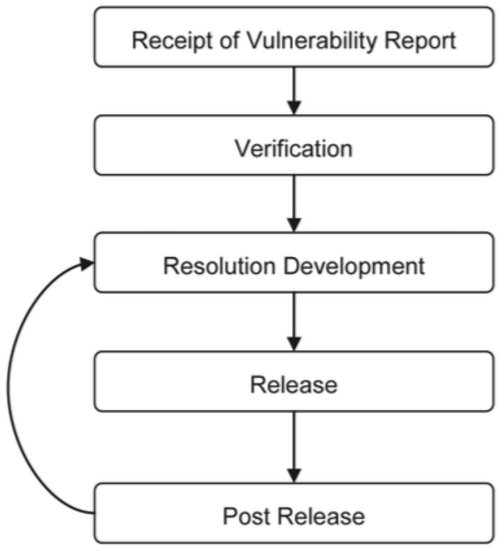


Figure 3.4: ISO 29147 Process Summary[65]

The ISO 29147 standard[65] requires the creation of a vulnerability disclosure policy, which states the intention and responsibilities of the vendor and vulnerability finder. A vulnerability disclosure policy should be clear and simple, to allow easy reporting of vulnerabilities. According to the ISO 29147 standard[65], the minimum aspects that a vulnerability disclosure policy should cover are:

- how the vendor would like to be contacted (email, phone, web form).
- secure communication options and expectations.
- useful information when reporting a vulnerability.
- out of scope services.
- how submitted reports are tracked.

There are also optional aspects such as crediting the vulnerability finder, public disclosure and distribution of a security advisory. It is recommended to publish this vulnerability disclosure policy, although parts of it with sensitive content can be kept internal to the organization[65].

For receiving vulnerability information, the ISO 29147 standard[65] details the information that should be included. A typical report includes a description of the vulnerability, proof of concept code, and the target product which is vulnerable. Because vulnerability reports often include sensitive information, a vendor should provide a means for secure transmission. Once a vulnerability report has been received, the ISO 29147 standard[65] suggests responding to the finder within the time period defined in the vulnerability disclosure policy. Communication should be ongoing with the finder, until the vulnerability is resolved. Finally, the ISO 29147 standard[65] requires having a tracking system to record all incoming vulnerability reports. A unique identifier should be assigned to the report, and used in further communication with stakeholders[65].

The ISO 29147 standard[65] also provides guidelines for the dissemination of an advisory. This is done after the presence of a vulnerability is confirmed, and information is prepared to help affected users. According to the ISO 29147 standard[65], a number of things need to be considered when creating a process for publishing advisories. First, the intended audience should be considered from a reader perspective, stating the target of the advisory. A method for verifying the authenticity and integrity of the advisory should be provided, to avoid counterfeit. Next, an advisory should ideally be accompanied with the remediation of the vulnerability. In situations where this is not possible, workarounds should be given. Further, if a vulnerability affects multiple vendors, then an advisory is best released by coordinating all parties involved. Vendors should also consider creating a mailing list for interested parties, and provide a scoring system such as CVSS for disclosed vulnerabilities. Finally, a trusted database with information about remediated vulnerabilities should be created and offered to both public and private consumers of vulnerability information[65].

The contents of an advisory are also specified by the ISO 29147 standard[65]. Typically this includes an identifier, a title, and overview, affected products, a description, impact, remediation, references, credits, a revision history, contact information and terms of use. In order to improve understanding of an advisory, the ISO 29147 standard[65] suggests using a consistent format for published advisories.

3.4 NIST 800-53

The NIST 800-53 special publication[42] is a publication which provides a catalog of security controls and a process for selecting them, to protect assets, operations and data from threats. The first version came out in 2005[17], and since then there have been 4 revisions. There is a 5th revision[18] under way, which is currently in final draft state.

According to Dempsey et al.[68], this publication provides a comprehensive set of security controls, three baselines (low, moderate, and high), and guidance for selecting the appropriate security controls. They argue that the provided security controls are designed to be technology-neutral, and focus on the fundamental measures needed to protect the organization. The NIST 800-53 special publication[42] is composed of the following parts:

1. **Introduction**: provides purpose, applicability, target audience and relationship to other publications.
2. **The fundamentals**: provides risk management approach, security control structure and baselines.

[17] SP 800-53, https://csrc.nist.gov/publications/detail/sp/800-53/archive/2005-02-28, last visited on 05/24/2021
[18] SP 800-53 Rev. 5, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final, last visited on 05/24/2021

3. **The process**: provides a process for selecting security controls, tailoring and documenting the selection process.

According to NIST's 800-53 special publication[42], the purpose of this publication is to provide guidelines and security controls, for meeting the minimum security requirements for federal information and information systems[19]. It applies to all components of information systems that store, process or transmit federal information. With the exception of national security systems, this publication applies to all federal information systems. The publication's intended audience is information security professionals, but also individuals with system development responsibilities, and commercial companies producing security-related technologies.

As shown by Dempsey et al.[68], to integrate a risk management process throughout an organization, the NIST 800-53 special publication[42] provides a three-tiered approach. Figure 3.5 shows a summary of this process. By dividing the risk management process into layers, risks can

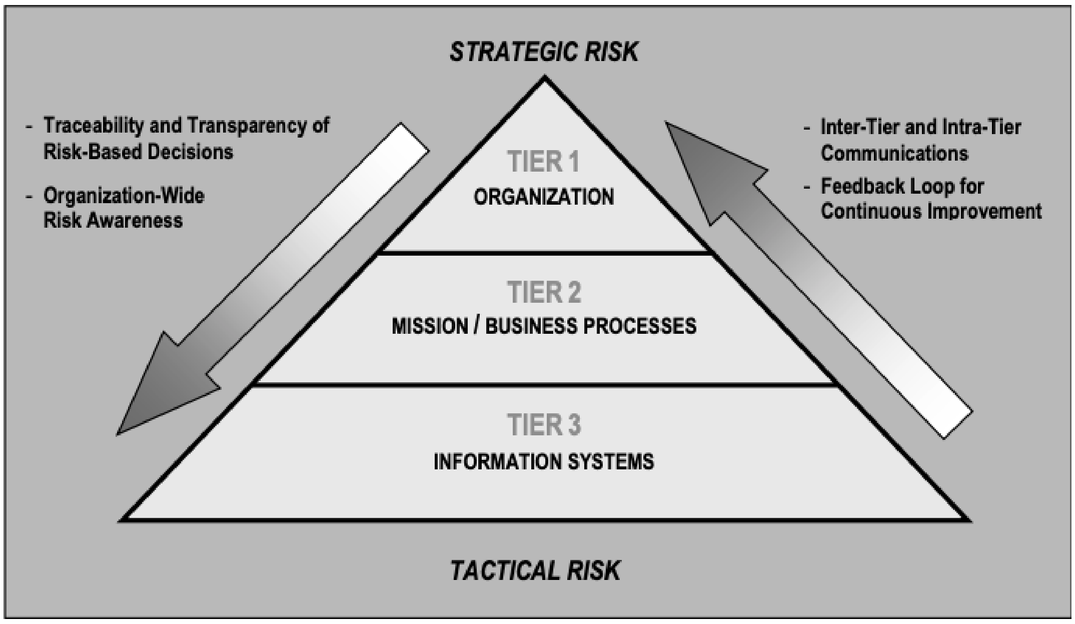

STRATEGIC RISK
- Traceability and Transparency of Risk-Based Decisions
- Organization-Wide Risk Awareness
TIER 1
ORGANIZATION
TIER 2
MISSION / BUSINESS PROCESSES
TIER 3
INFORMATION SYSTEMS
- Inter-Tier and Intra-Tier Communications
- Feedback Loop for Continuous Improvement
TACTICAL RISK

Figure 3.5: NIST 800-53 special publication risk management approach[42]

be addressed at the organizational level, mission / business process level and information systems level. The overall objective remains the continuous improvement of the organization's security posture, and a feedback loop is maintained through through inter-tier communication.

The security controls provided by NIST's 800-53 special publication[42] have a well defined structure, and are organized in families. Each family contains security controls related to a specific security topic. Table 3.2 shows the security control identifiers and family names. The security controls within these families treat various aspects of an organization's security program, including policy, oversight, supervision, manual and automated processes, individual actions etc.

To overcome the challenge of selecting the right set of security controls, NIST's 800-53 special publication[42] provides the concept of a baseline. A baseline represents the starting point for the security control selection process, and is based on the security category and impact level

[19] FIPS 200, https://csrc.nist.gov/publications/detail/fips/200/final, last visited on 05/24/2021

of information systems. In this special publication, there are three security control baselines: low-impact, moderate-impact, and high-impact baselines for information systems. These align with the three security categories provided by FIPS 200[19], which are used to determine the impact level of systems.

ID	**FAMILY**	**ID**	**FAMILY**
AC	Access Control	MP	Media Protection
AT	Awareness and Training	PE	Physical and Environmental Protection
AU	Audit and Accountability	PL	Planning
CA	Security Assessment and Authorization	PS	Personnel Security
CM	Configuration Management	RA	Risk Assessment
CP	Contingency Planning	SA	System and Services Acquisition
IA	Identification and Authentication	SC	System and Communications Protection
IR	Incident Response	SI	System and Information Integrity
MA	Maintenance	PM	Program Management

Table 3.2: NIST security controls and identifiers[42]

As shown by Dempsey et al.[68], organizations may tailor the selected security control baseline in any of the three tiers, to support particular security objectives. The process of tailoring the baseline is comprised of several steps, as shown in NIST's 800-53 special publication[42]:

1. Identifying and designating common controls in initial security control baselines.
2. Applying scoping considerations to the remaining baseline security controls.
3. Selecting compensating security controls, if needed.
4. Assigning specific values to organization-defined security control parameters via explicit assignment and selection statements.
5. Supplementing baselines with additional security controls and control enhancements, if needed.
6. Providing additional specification information for control implementation, if needed.

Once the appropriate security controls are selected and adapted for the organization use case, NIST's 800-53 special publication[42] requires the documentation of the process. Dempsey et al.[68] argue that this documentation aids in review activities, security planning and future risk assessments. According to Dempsey et al.[68], this documentation is essential when examining security considerations for the information systems of an organization.

3.5 TIBER-EU

The European framework for „threat intelligence-based ethical red-teaming“(TIBER)[54] is an EU-wide guide for testing the resilience of entities against threats. It represents an intelligence-based, controlled red team test of a target's critical production systems. Such a test involves the use of various real-life attacking techniques to simulate sophisticated cyber attacks.

TIBER-EU has been developed by the European Central Bank and EU national banks, published in May 2018[20]. The main objectives of this framework are the enhancement of entities' resilience (especially the finance sector) against cyber attacks, the standardization of red team testing across the EU, providing guidance for establishing this form of testing in the European level, and enabling collaboration of the various national authorities. The adoption of TIBER-EU is voluntary and can be done at a national or european level[54]. Typical authorities involved in the implementation are: central banks, supervisory authorities, intelligence agencies, relevant ministries, etc.

The TIBER-EU[54] test process is composed of three mandatory phases:

1. Preparation phase
2. Testing phase
3. Closure phase

Figure 3.6 shows each of these phases and the procedures within them.

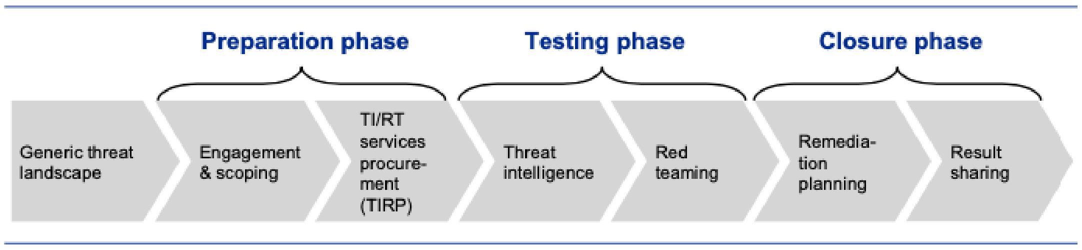


Figure 3.6: TIBER-EU[54] test process

Preparation Phase In the preparation phase, the TIBER-EU test engagement is launched formally. This includes a number of activities. First, the entities which are going to perform the TIBER-EU test should be decided. The relevant authorities are then notified in a pre-launch meeting. What follows is the procurement of services for external threat intelligence and red teaming services. There is a formal TIBER-EU certification and accreditation process, which helps to identify suitable providers of these services. Finally, the engagement is formally scoped between the entities and relevant authorities involved. This concludes the preparation phase[54].

Testing Phase After scoping has been completed, TIBER-EU[54] explains that the testing phase begins. This is composed of two parts: threat intelligence and red teaming. Threat intelligence aims to use reconnaissance and related actions that help develop attack scenarios. The result of the threat intelligence procedure is a targeted threat intelligence (TTI) report[54]. In the red-teaming procedure, the service provider executes a targeted attack against the systems, based on the TIBER-EU[54] red team testing guidelines. The provider conducts a realistic and comprehensive red team test based on the objectives agreed in the preparation phase. The testing methodology is an emulation of real world attacks as described in section 2.2.3.

Closure Phase This is the final phase and includes remediation planning and result sharing[54]. The service provider will deliver a red team test report which includes details of the testing process and findings. From there, a remediation plan is developed by the entities responsible as defined in

[20] What is TIBER-EU?, https://www.ecb.europa.eu/paym/cyber-resilience/tiber-eu/html/index.en.html, last visited on 05/24/2021

the preparation phase. This remediation plan can be used to present a business case for patching the vulnerabilities identified during the red team test. Once everything in the reports has been agreed upon, the results can be shared with other relevant authorities[54].

TIBER-EU[54] also suggests the creation of a „TIBER Cyber Team"(TCT) that will monitor the process of implementation and maintenance. This team is also responsible for the necessary modifications when adopting the framework at a national level. A TCT can be set up as a single authority, or a centralised team of experts from different parts of authorities[54].

3.6 Austrian Information Security Manual

The austrian information security manual[55] is a joint development of the austrian federal chancellery and austrian center for secure information technology (A-SIT)[21], in cooperation with the german federal office for security in information technology[22] and swiss federal IT steering unit[23]. It made its first appearance in October 1998[24], and the current release dates December 2019[55].

The main purpose of the austrian information security manual[55] is to help public and private organisations in implementing the ISO 27000 family of standards[25]. Additionally, this manual fills the void of a comprehensive and detailed guideline for information security. It is also considered good material for furthering education, and as a helping tool for self-checks[55].

The structure is closely aligned with the ISO 27001 and 27002 standards, with 18 chapters and a number of appendixes[55]. Chapter one provides an introduction to the manual, and an executive summary. In chapter two and three, the process of maintaining and improving information security is explained. This is primarily directed at public organizations, but the same can be used for private organizations as well. Chapter 4 to 18 describe the details for correctly implementing security measures in various fields such as personnel, infrastructure, technology etc. Finally, the appendixes provide additional helping material such as documentation templates, references and tools[55].

According to the austrian information security manual[55], information security management is a continuous process where effectiveness and strategy must be constantly measured and improved. Core activities as part of an information management system include: development of information security policies, performing risk assessments, creating a security concept, implementation of security measures, ensuring information security in production environments, and constant monitoring and improvement of the information security management system. This process is shown in figure 3.7 and can be applied to parts or the whole organization.

Development of Information Security Policies As shown in the austrian information security manual[55], an organization-wide information security policy provides a long-term strategy for securing information systems. It is part of a hierarchy of documents and depends upon the security requirements and IT infrastructure of the organization. A detailed description of security

[21] Austrian Center for Secure Information Technology, https://www.a-sit.at/en/secure-information-technology-center-austria, last visited on 05/24/2021

[22] German Federal Office for Security in Information Technology, https://www.bsi.bund.de/EN/Home/home_node.html, last visited on 05/24/2021

[23] Federal IT Steering Unit, https://www.isb.admin.ch/isb/en/home.html, last visited on 05/24/2021

[24] Österreichisches Informationssicherheitshandbuch – A-SIT. https://sectank.net/blog/2011/03/25/osterreichisches-informationssicherheitshandbuch, last visited on 05/24/2021

[25] Family of IT Security Standards, https://www.iso.org/search.html?q=27000, last visited on 05/24/2021

policies is provided in section 3.1.1.

Performing Risk Assessments The austrian information security manual[55] provides three risk assessment strategies as part of its information security management system (ISMS) guideline: detailed risk assessment, base assessment and a combined assessment. The selection of the appropriate risk assessment is left to the organization, and should be specified in the information security policy. Further details on risks and the risk assessment process can be found in section 2.1.5

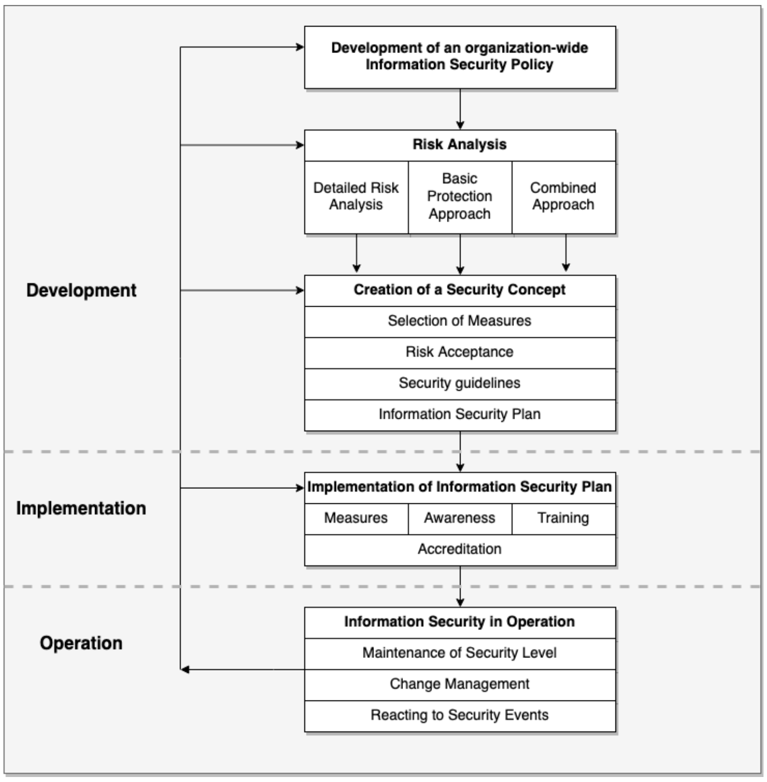

Development of an organization-wide Information Security Policy
Risk Analysis
Detailed Risk Analysis
Basic Protection Approach
Combined Approach
Development
Creation of a Security Concept
Selection of Measures
Risk Acceptance
Security guidelines
Information Security Plan
Implementation
Implementation of Information Security Plan
Measures
Awareness
Training
Accreditation
Operation
Information Security in Operation
Maintenance of Security Level
Change Management
Reacting to Security Events

Figure 3.7: Austrian Information Security Manual ISMS process[55]

Creating a Security Concept According to the austrian information security manual[55], separate security guidelines should be created for important IT systems and applications. These provide the concrete measures taken to secure those systems, and should be compatible with the organization's information security policies. The selection of these measures, together with the creation of security guidelines and risk assessments, provide the elements of a security concept.

Implementation of Security Measures Once the appropriate measures have been selected, according to the austrian information security manual[55], they can be implemented. For correct and efficient implementation, the financial and personnel responsibilities should be defined. Additionally, project management techniques should be used to ensure expected timelines and costs are met. Simultaneous with the implementation of security measures, users should be trained and prepared for correct usage[55].

Information Security in Production Environments After the desired security level has been reached, the austrian information security manual[55] explains that maintenance is required to maintain that level of security. Because the implemented security measures will lose effectiveness over time (due to new threats and risks), further activities need to counteract and maintain the security of information systems.

Monitoring and Improvement of the Information Security Management System
Similar to the previous activity, the information security management system must be maintained and improved. The austrian information security manual[55] provides guidelines for implementing monitoring where feasible. Through monitoring, actual metrics can be compared to expected values and as such, the organization can react to deficiencies in the security of information systems, or react to an attack. Monitoring also allows to provide a complete picture to upper management, showing the achieved security level, measures taken, and risks mitigated.

3.7 Discussion

As shown in the previous sections, there are many resources to help organizations manage their IT Security strategy. They provide various aspects of information security management, such as the Plan-Do-Check-Act security process of ISO 27001[49], NIST's risk management framework[42], TIBER-EU's framework for red team assessments[54], and the ISMS process of the austrian information security manual[55]. For the purpose of this thesis, the security controls and offensive security aspects of these resources are of interest.

In terms of offensive security, the ISO 27001[49] standard provides the A.12.6 control called „Technical vulnerability management", which requires the detection and addressing of technical vulnerabilities. However, this control does not specify *how* vulnerability information is obtained. It is left to the organization to decide the appropriate means. Similarly, NIST's[42] special publication provides the „Security Assessment and Authorization" family of controls for offensive security. This control requires the use of security assessments and penetration testing for offensive security. Penetration tests are also part of the austrian information security manual[55], which also requires security tests, vulnerability identification and qualification tests. TIBER-EU[54] provides a framework for red team assessments as part of offensive security, whereas ISO 29147 [65] only describes the intake of vulnerabilities from public sources. Table 3.3 shows the comparison of offensive security methods provided by these resources.

Resource	**Offensive Security Methods**
ISO 27001	Left at discretion of organization
NIST	Security Assessments Penetration Testing
Austrian Information Security Handbook	Security Tests Penetration Testing Vulnerability Identification Qualification Tests
TIBER-EU	Red Team Assessment
ISO 29147	Vulnerability intake from public sources

Table 3.3: Comparison of offensive security methods

As shown, none of the resources cover a crowdsourced approach for offensive security. Although case studies and reports show promising results of using crowdsourced security to increase the number of vulnerability reports[17], there is a lack of adoption by enterprise organizations[9]. A possible cause for this could be missing policies and integration concepts for this new offensive security measure, and the lack of standards that support it.

4 Bug Bounties: The Crowdsourced Offensive Security Approach

Bug bounties are an offensive security method, where crowdsource techniques are used to invite participants to find security vulnerabilities. These participants are commonly known as whitehat hackers or security researchers[69]. Typically, finding a vulnerability and reporting it to the vendor results in a monetary reward, acknowledgement, or some other form of recognition.

According to Fryer and Simperl[10], bug bounty programs have become a popular cybersecurity initiative in recent years. This kind of approach has seen adoption in many different areas such as e-voting systems[1], government systems[2] and self-driving cars[3]. As a result, platforms that offer tools and services for managing bug bounties have appeared, such as Bugcrowd and HackerOne. According to Bugcrowd's annual report[70], an increase of 29% in the number of programs launched has been observed in 2019. Additionally, a 50% increase in public programs was also noted.

4.1 Characteristics

In the following section, bug bounty definitions and characteristics are laid out. First, a definition of bug bounty itself is given, followed by a definition for the crowd. The different bug bounty types and their usage are covered next. Finally, the various aspects that must be considered for running a bug bounty program are described in detail.

4.1.1 Definitions

One definition of bug bounty programs is given by Kuhen and Mueller[17]:

> *„Bug bounty programs (BBP), also referred to as vulnerability rewards programs (VRP) or bug challenges, reward independent security researchers, penetrations testers, and whitehat hackers for discovering exploitable software vulnerabilities and sharing this knowledge with the operator of a particular bug bounty program“.*

Another definition is given by Fryer and Simperl[10], who use the following:

> *„A 'bug bounty' or 'vulnerability reward program' (VRP) is the process for rewarding the discovery of a flaw or vulnerability in a piece of software.“.*

[1] Switzerland paying e-voting hackers, https://www.euronews.com/2019/02/13/switzerland-offers-cash-to-hackers-who-can-crack-its-e-voting-system, last visited on 05/24/2021

[2] United States Department of Defense: Expanding hack the Pentagon, https://dod.defense.gov/News/News-Releases/News-Release-View/Article/1671231/department-of-defense-expands-hack-the-pentagon-crowdsourced-digital-defense-pr, last visited on 05/24/2021

[3] AI trends: Bug bounties and AI systems: The case of AI self-driving cars, https://www.aitrends.com/ai-insider/bug-bounties-and-ai-systems-the-case-of-ai-self-driving-cars, last visited on 05/24/2021

As stated above, the essence of bug bounty programs is rewarding the finder for responsibly sharing the vulnerability information with the vendor. For the purpose of this thesis, the definition given by Fryer and Simperl[10] will be used, as it sufficiently describes the nature of bug bounties.

The finder is defined as any independent security researcher that is able to identify security vulnerabilities in assets of the bug bounty vendor, and complies with the rules of engagement. In their recent report[71], Bugcrowd provides several insights into „the crowd“. According to this report, 46% of the crowd are of Asian ethnic and 24% are caucasian. The average age of security researchers is dominated by younger individuals aged 18-24 (53%), followed by 25-34 (32%). There are also various motivations for participating in bug bounties, such as improving skills, increasing security, fame and financial incentives.

4.1.2 Types and Usage

Depending on the method of implementation, bug bounties can be categorized in different types. One way to categorize bug bounties is given by Malladi and Subramanian[14], who use the following types:

1. Institutional bug bounty programs
2. Bug bounty platforms
3. Private intermediary programs

Institutional bug bounty programs are hosted directly by the bug bounty owner. Examples of these include Google's VRP[4], Facebook's bug bounty program[5], Microsoft bug bounty program[6] etc. In institutional bug bounty programs, the program operator controls the complete process, from publication of the rules of engagement to accepting and rewarding a vulnerability report. Bug bounty platforms are intermediary hosts of bug bounty programs for many organizations. They offer bug bounties as a service, and manage various aspects of the bug bounty program for their clients. Examples of prominent bug bounty platforms include Bugcrowd[7], HackerOne[8], Synack[9] etc. Private intermediary programs such as Zerodium[10] purchase vulnerabilities from security researchers and use them for their services. They are usually interested in novel research for popular products and provide higher rewards. For the purpose of this thesis, private intermediaries will not be considered as they violate the definition used for bug bounties.

The emergence of bug bounty platforms in the last years and their innovation has led to various methods of running a bug bounty program. For example, both Bugcrowd and HackerOne offer two types of bug bounty programs: public and private. A private bug bounty program represents an invite-only form of participation. In this type, the bug bounty platform will invite only a limited number of security researchers to the program. In public bug bounty programs, any researcher can participate. In both cases, all characteristics described in section 4.1 apply, and researchers must follow the rules of engagement to be eligible for a reward. The Bugcrowd bug bounty platform

[4] Google Vulnerability Reward Program, https://www.google.com/about/appsecurity/reward-program, last visited on 05/24/2021
[5] Facebook whitehat, https://www.facebook.com/whitehat, last visited on 05/24/2021
[6] Microsoft bug bounty program, https://www.microsoft.com/en-us/msrc/bounty, last visited on 05/24/2021
[7] Bugcrowd, https://www.bugcrowd.com, last visited on 05/24/2021
[8] HackerOne, https://www.HackerOne.com, last visited on 05/24/2021
[9] Synack, https://www.synack.com, last visited on 05/24/2021
[10] Zerodium, https://www.zerodium.com, last visited on 05/24/2021

offers additional program types, such as the three options for vulnerability disclosure programs[11]. These options include: email-intake, embedded form, and Bugcrowd hosted. Email intake allows organizations to configure an email address which can be used by security researchers to report vulnerabilities. In the embedded form option, a form for submitting vulnerability reports is published on the organization website. Both options result in vulnerability reports created within the Bugcrowd platform, which then are processed using the standard flow described in section 4.1. The final option is to run the program directly on the Bugcrowd platform, where the details and the processes are handled in it. A detailed description of bug bounty platforms is provided in section 4.4.

According to Malladi and Subramanian[14], crowd collaboration through bug bounties can be used at every phase of product development. For example, a fuzzing competition through a private bug bounty program can be used during the development phase. Fuzzing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to provoke errors or unexpected outcomes[38]. A description of a private bug bounty program is given in section 4.2. In summary Malladi and Subramanian[14] recommend conducting bug bounty engagements throughout the software development life-cycle as follows:

- Requirements and Design: Test plan and budgeting for bug bounty
- Development: Fuzzing competition through private bug bounty
- Public and Private Beta: Private bug bounty
- Product Market Launch: Either public or private bug bounty
- Post-Release: Public or private or fuzzing competition bug bounty

4.1.3 Running a Bug Bounty Program

As shown by Bell et al.[72], running a bug bounty program requires careful consideration of a number of aspects such as: staffing, rules of engagement, rewards, processing reports, public disclosure.

Staffing Bug bounties require a solid security team to handle the incoming vulnerability reports. Being able to quickly respond to vulnerability reports and fostering relationships with the bug bounty community is a key factor to the success of the program. Using a bug bounty platform can help acting as a filter to reduce the number of invalid reports, however it comes at the cost of having less control over the program. The benefits and drawbacks of using a bug bounty platform are elaborated further in section 5.2

Rules of engagement As shown by Laszka et al.[73], the rules of engagement define the program rules that govern interactions between organizations and whitehat hackers. Common items that are defined in the rules of engagement include: in-scope targets, expectations around rewards, eligible vulnerability types, legal requirements etc. The „in-scope targets“ section defines specifically which targets should be tested. There may also be a specific list of targets explicitly out-of-scope and for which testing is prohibited[73]. The rules of engagement also specify expectations for the type of reward received when reporting a valid vulnerability. Additionally, it is common to exclude a number of vulnerability types which don't apply to the application. This is

[11] Bugcrowd vulnerability disclosure program, https://www.bugcrowd.com/products/vulnerability-disclosure, last visited on 05/24/2021

done by either including a list of eligible vulnerability types, or an exclusion list. Finally, the rules of engagement typically specify legal information which permit the testing of listed targets[72].

Rewards As shown by Bell et al.[72], it is good practice to state the type of reward that security researchers can expect for reporting a vulnerability. For example, a reward can be a sum of money, items such a t-shirts or stickers, or public acknowledgement. For monetary rewards, it is common to have reward ranges associated with the severity of the vulnerability, and potentially bonuses for high impact. Delivering the money can be tricky considering that researchers may be all over the world. Bug bounty platforms help in this regard by taking that responsibility of issuing the reward to the security researcher. Similar logistical challenges are expected when items such as t-shirts or stickers are rewarded. Another common form of reward is listing the security researcher in a bug bounty hall of fame[12]. Kuehn and Mueller[17] argue that reputation is a crucial element in the security research community, and being listed in a popular bug bounty hall of fame will increase a security researcher's reputation.

Processing reports Once a vulnerability has been identified and reported by a security researcher, the bug bounty program operator must process it. The typical flow of a vulnerability report as part of a bug bounty program is shown in figure 4.1. Initially, the vulnerability report is checked to ensure it complies with the rules of engagement. If the vulnerability report is not compliant, the researcher is notified and no further processing is performed. Typically, bug bounty platforms apply penalties[13] for reports that do not comply with the rules of engagement, as a means to drive down the number of invalid reports. Next, the vulnerability report is compared against previous reports to determine whether it is a duplicate. Bug bounties work by only rewarding the first finder of a vulnerability. All future reports that have the same root cause are considered duplicates and receive no reward[14]. Provided that both of these checks are cleared, typically the bug bounty operator attempts to reproduce the behaviour internally. As shown by Malladi and Subramanian[14], this is an important step because of quality issues that arise from misaligned researcher motivations. Reports may miss critical information, which requires the bug bounty operator to ask clarifying questions and invoke the cycle of communication as shown in figure 4.1. To avoid this situation, Bell et al.[72] suggest creating a structure in the vulnerability report form used by security researchers. This has also further advantages such as better metrics and keeping vulnerability reports separated[72]. Finally, if the report is a valid security vulnerability, it can be filed with the organization's internal systems used to keep track of vulnerabilities. At this point, the researcher is notified that they have reported a valid security vulnerability, and a reward is issued.

Public disclosure The last aspect of running a bug bounty program is public disclosure. Similar to the motivations for having a hall of fame mentioned above, security researchers typically like to publicly disclose their report. This is another means for the security researcher to increase their reputation. However, Laszka et al.[73] argue that public disclosure is a concern for the bug bounty operator. Because publicly disclosing vulnerability reports gives information about the internal security state of an organization, a bug bounty program may prohibit disclosure by specifying so in the rules of engagement.

[12] Firefox Bug Bounty Rewards, https://www.mozilla.org/en-US/security/bug-bounty/hall-of-fame, last visited on 05/24/2021

[13] The Importance of Scope – Bug Bounty Hunter Methodology, https://www.bugcrowd.com/blog/the-importance-of-scope-bug-bounty-hunter-methodology, last visited on 05/24/2021

[14] Duplicate Reports, https://docs.HackerOne.com/programs/duplicate-reports.html, last visited on 05/24/2021

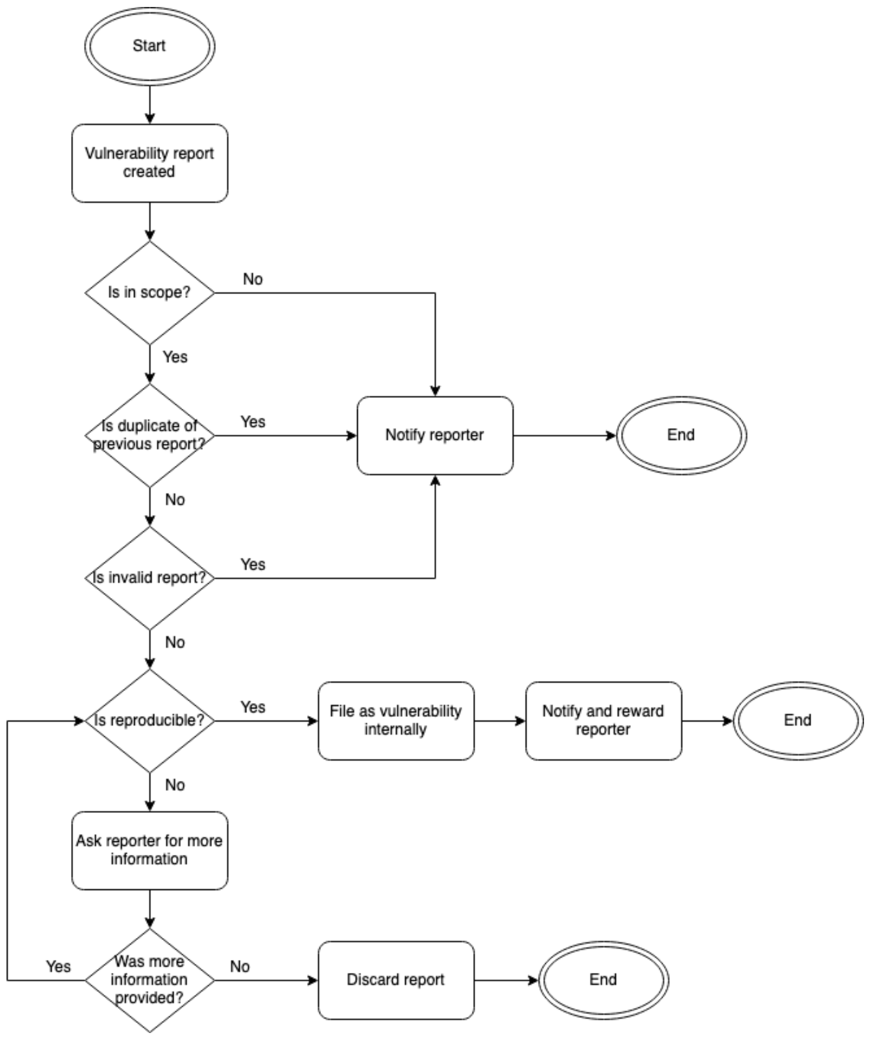

Start
Vulnerability report created
Is in scope?
No
Yes
Is duplicate of previous report?
Yes
Notify reporter
End
No
Is invalid report?
Yes
No
Is reproducible?
Yes
File as vulnerability internally
Notify and reward reporter
End
No
Ask reporter for more information
Yes
Was more information provided?
No
Discard report
End

Figure 4.1: Vulnerability report flow as defined for this thesis.

In their study, Laszka et al.[73] found that 38 out of 111 programs had information about public disclosure in their rules of engagement. However, as shown in section 2.1.3, disclosure of vulnerabilities is a response against the insecurity of software and hardware that attempts to provide incentives for faster patching[35]. Hence, public disclosure of vulnerabilities reported through bug bounty programs is generally desired. Popular platforms such as HackerOne[15] and Bugcrowd[16] offer built-in functionality for researchers to request public disclosure, and for bug bounty operators to approve or deny them.

4.2 Metrics

As shown in section 1.1, there are a set of pre-adoption fears and post-adoption issues that prevent some organizations from using crouwdsourced security. Al-banna et al.[8] show that one of these post-adoption issues is the low quality of submissions. Hence, quality of vulnerability reports represents one of key metrics for measuring the effectiveness of a bug bounty program.

[15] Disclosure, https://docs.HackerOne.com/hackers/disclosure.html, last visited on 05/24/2021
[16] Public disclosure policy, https://researcherdocs.bugcrowd.com/docs/disclosure, last visited on 05/24/2021

In this section, a number of metrics will be defined, that allow to measure and improve the success of a bug bounty program. Quantitative analysis will look at the number of valid and invalid reports. This is commonly referred to as signal to noise ratio[8] . Other metrics such as the quality of vulnerability reports with regards to reproducibility, severity and vulnerability types are also covered.

4.2.1 Quantitative Analysis

The most common quantitative analysis in bug bounty programs is the signal to noise ratio. As shown by Al-banna et al.[8], this represents the ratio of valid and invalid vulnerabilities compared to the total number of reports. Bugcrowd shows that in 2018, 30% of submissions were invalid and accounted for noise[74]. HackerOne reported a platform-wide signal of 80% for 2018, which means that 20% of vulnerability reports were noise[17].

According to Laszka et al.[75], invalid reports are considered any type of report that is either spam (completely irrelevant), false positive (issues that don't exist or do not have security impact), or reports that do not comply with the rules of engagement. Typically, invalid reports are a result of incorrect research and lack of validation by the finder. For example, a security researcher might run a vulnerability scanner and report the findings without manual verification. Since vulnerability scanners commonly report false positives[76], these reports usually have an invalid outcome during triage. Other examples of invalid reports include submissions where the security researcher makes up false assumptions about the target application. Since filtering out noise is costly for a bug bounty program operator, they are usually accompanied by some form of penalty. For example, Bugcrowd applies a -1 point penalty for out of scope submissions[18], and HackerOne applies -5 points for not applicable and -10 points for spam reports[19]. More aggressive penalties may drive a better signal to noise ratio, but could also result in fewer valid submissions.

Another common form of avoiding a high amount of noise, is to run a private bug bounty program. As established in section 4.2, a private bug bounty program invites a limited number of participants. The participants can be chosen based on their previous records of valid reports. In figure 4.2, Zhao et al.[15] show the results of public vs private programs from data obtained by the HackerOne bug bounty platform. As is evident, the amount of noise in private programs is considerably less than in public programs. However, Zhao et al.[15] argue that due to the limitations in using the crowd, private programs cannot produce the same results as public programs can. Hence, the total number of valid reports is higher for public bug bounty programs.

4.2.2 Qualitative Analysis

In the qualitative analysis of vulnerability reports, the ability to clearly articulate the security problem is analyzed. This includes describing the vulnerability root cause, providing exact reproduction steps and potentially a method for patching. As shown by Zhao et al.[15], failing to clearly explain the vulnerability not only leads to many rounds of communication between the whitehat hacker and bug bounty operator, but can also result in the report being marked invalid. This process is costly for the bug bounty operator, and can result in delays while issuing a reward

[17] The Hacker-Powered Security Report 2018, https://www.HackerOne.com/blog/118-Fascinating-Facts-HackerOnes-Hacker-Powered-Security-Report-2018, last visited on 05/24/2021

[18] Researcher Documentation - Reporting a bug, https://researcherdocs.bugcrowd.com/docs/reporting-a-bug, last visited on 05/24/2021

[19] Report States, https://docs.HackerOne.com/programs/report-states.html, last visited on 05/24/2021

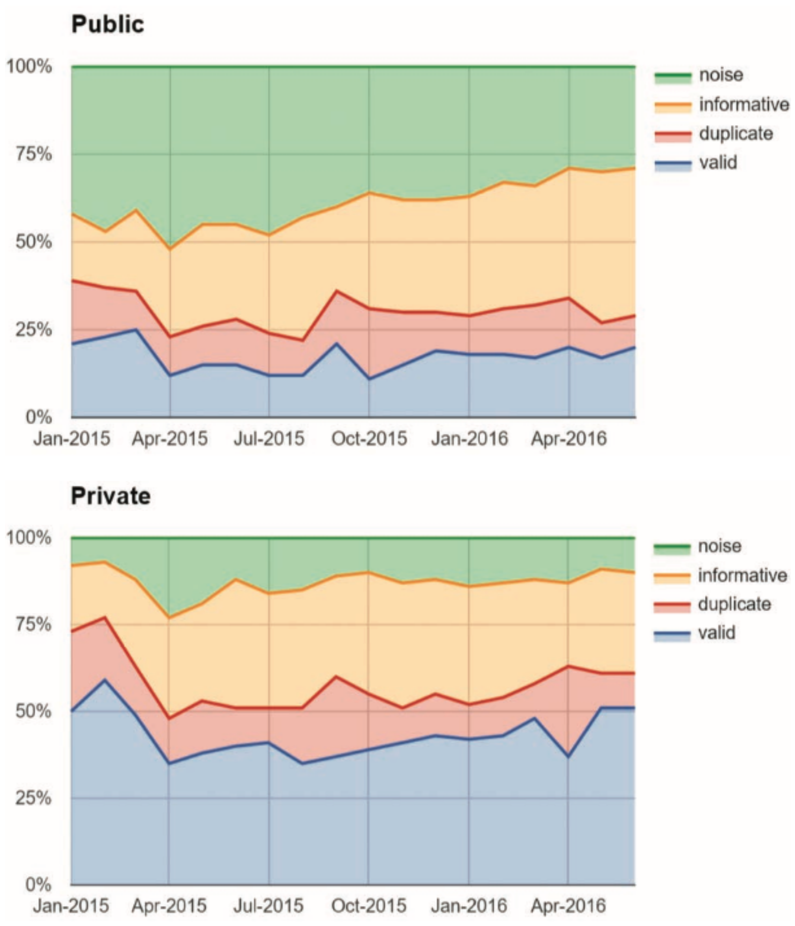


Figure 4.2: Noise in public and private programs[15]

to the researcher.

According to Malladi and Subramanian[14], a pre-requisite for high-quality vulnerability reports is the provision of a „proof of concept“ test case. This allows the bug bounty operator to see the vulnerability in action, and proves its existence. Similar arguments are made by Zimmerman et al.[77], who report that steps to reproduce and stack traces are the most useful information in bug reports. Although Zimmerman's study is not restricted to vulnerability reports, the same principles apply. They found that the most severe problems encountered by developers are errors in steps to reproduce, incomplete information, and wrong observed behaviour[77]. The results of these problems were longer times to triage the bug report, and false positives.

There are various ways to incentivize high-quality vulnerability reports. Zhao et al.[15] suggest that instructions in the bug bounty rules of engagement can improve the quality of reports submitted by whitehat hackers. By clearing stating reporting guidelines and vulnerability types of interest, organizations can direct finders efforts and achieve desired results. A similar argument is given by Munaiha and Meneely[5], who noticed that overall report quality and reproducibility increased when emphasized in the bounty criteria. An additional motivator for quality vulnerability reports is higher rewards, as shown by Malladi and Subramanian[14]. This plays into the financial incentive that is a primary driver for many whitehat hackers. By attaching the reward amount to the quality of the vulnerability report, an increase of quality can be expected (although at a higher cost). The same has been confirmed by Laszka et al.[75], who show that the number of invalid reports can be reduced by rewarding the researcher validation efforts. Since writing a good and detailed vulnerability report requires effort, rewarding such behaviour can be traced to an increase in vulnerability report quality.

4.2.3 Vulnerability Types and Severity

Since bug bounties utilize a large crowd that typically has a variety of skillsets, many types of vulnerabilities are uncovered. In their empirical analysis, Zhao et al.[12] show a correlation between the size of the finder community, and the number and types of vulnerabilities found. They also show that the top 3 vulnerability types were cross-site scripting (XSS), sql injection and logic errors. Although cross site scripting remains the most common security problem, other top categories have changed as shown in table 4.1. This table shows the top vulnerability types reported in the Bugcrowd[70] and HackerOne[9] bug bounty platforms.

No	Bugcrowd	HackerOne
1	Cross-Site Scripting (XSS)	Cross-Site Scripting (XSS)
2	Improper Access Control	Information Disclosure
3	Privilege Escalation	Violation of Secure Design Principles
4	Open redirect	Improper Authentication
5	Cross-Site Request Forgery (CSRF)	Cross-Site Request Forgery (CSRF)
6	Broken Authentication & Session Management	Open Redirect

Table 4.1: Comparison of vulnerability types on Bugcrowd and HackerOne

As shown in section 2.1.3, there are a number of ways to classify vulnerabilities. While HackerOne uses a subset of the common weakness enumeration[20] classification, Bugcrowd have created their own classification named „Bugcrowd Vulnerabilty Rating Taxonomy“[21]. Whichever method of classification is used, bug bounties typically require a means to identify the type of vulnerability that is being reported. This helps with further actions such as triaging the vulnerability, rewarding, remediation etc.

Similar to vulnerability type, the vulnerability severity is a critical component in bug bounties. Generally, the amount of reward is closely tied to the severity: the higher the severity, the higher the reward. For example, Bugcrowd[22] rewards 40 reputation points for a critical vulnerability and HackerOne offers 50 points for the same[23]. While monetary rewards can vary across programs, the principle of higher severity = higher reward remains. Vulnerabilities with critical impact can earn tens of thousands of dollars, as shown by the largest bounty ever paid by Facebook[24].

One way to determine the severity of a vulnerability, is to use the CVSS[25] system as shown in section 2.1.3. HackerOne uses this method for determining the severity (and later the reward), while Bugcrowd uses their own system severity levels of P1(critical) to P5(informational).

[20] Weakness, https://docs.hackerone.com/hackers/weakness.html, last visited on 05/24/2021
[21] Bugcrowd's Vulnerability Rating Taxonomy, https://bugcrowd.com/vulnerability-rating-taxonomy, last visited on 05/24/2021
[22] Getting Rewarded, https://researcherdocs.bugcrowd.com/docs/getting-rewarded, last visited on 05/24/2021
[23] Effects of Bounties on Reputation, https://docs.hackerone.com/hackers/reputation.html, last visited on 05/24/2021
[24] Facebook, Under Scrutiny, Pays Out Largest Bug Bounty Yet, https://www.wired.com/story/facebook-bug-bounty-biggest-payout, last visited on 05/24/2021
[25] Common Vulnerability Scoring System, https://www.first.org/cvss, last visited on 05/24/2021

4.2.4 Reporting and Documentation

As shown in section in section 4.2.1, running a bug bounty program will result in overhead required to manage noise. Inherently, reporting is required to measure the performance of a bug bounty program. Apart from the quantitative metrics of section 4.2.1 and qualitative metrics of section 4.2.2, the following data points can be used to measure a bug bounty program:

1. Finder participation
2. Program response time
3. Program reward amounts
4. Coverage

1. Finder participation As shown by Zhao et al.[12], there exists a correlation between the number of whitehat participants and the number of vulnerabilities discovered. This means that the larger the crowd, the more vulnerabilities will be found. It can simply be measured as the number of people who have reported at least one vulnerability. For private bug bounty programs, a differentiation between the number of people who *accepted* an invitation and those who actually found a vulnerability can be shown. However, as explained by Zhao et al.[12], the productivity of finders is more important than just the plain number of reports. That is due to reports being duplicate or invalid. As such, the productivity of whitehat hackers (in the form of signal to noise)can be taken into account as well.

2. Program response time Another important metric for ensuring the success of a bug bounty program is response time. As shown by Zhao et al.[15], this is considered the time to the first response of vulnerability report, the time to triage it, and the time to issue a reward. A timely response to a vulnerability report ensures the researcher is promptly rewarded for their efforts, which in turns keeps them engaged and motivated. Bug bounty platforms go so far as to set requirements on responsiveness for their clients. For example, HackerOne[26] requires a maximum of 5 days response for first touch, a 10 days response for triage, and a reward timeline of 1 day after triage. According to Walshe and Simpson[69], this standard is met in 97% of vulnerability reports.

3. Program reward amounts The reward amounts give great insight in the return of investment for the bug bounty program. Additionally, they serve as a means to motivate and attract whitehat hackers, as shown in section 4.2.2. In their study, Walshe and Simpson[69] found that the average cost of running a bug bounty program for an entire year is less than hiring two additional software engineers. Additionally, they found that the average reward is $318, which can be used to determine the initial reward amounts based on severity. As the program continues, the reward amounts can be adjusted to maintain a stable cost of operation.

4. Coverage As shown by Großmann et al.[78], coverage is an important item for security testing. The same principle applies for bug bounty programs. To ensure crowd collaboration on all the in-scope targets, coverage must be consistently monitored and evaluated. One method to measure coverage is to require security researchers to use a VPN and monitor all traffic. This

[26] Response Standards and Targets, https://docs.hackerone.com/programs/response-target-metrics.html, last visited on 05/24/2021

is popular method and particularly popular with bug bounty platforms, as shown by Synack's[27] LaunchPoint technology which does exactly that.

4.3 Commercial Platforms

As mentioned in section 1.1, the popularity of crowdsourced offensive security resulted in the emergence of bug bounty platforms. These platforms act as an intermediary between the white-hat hackers and organizations running bug bounty programs. They offer various services such as clearing out the noise, managing the crowd, and integrations with external systems.

The biggest and oldest platforms are Bugcrowd and HackerOne, which have been mentioned throughout this chapter. At the time of writing this thesis, other platforms found offering bug bounty services include: Cobalt, Synack, Intigriti, Yeswehack, and Safehats.

Bugcrowd Bugcrowd was founded in 2012[28] and is the oldest of the platforms. At the time of this writing, Bugcrowd offers four major solutions[29]: Penetration Testing, Bug Bounty, Vulnerability Disclosure, and Attack Surface Management. Their penetration testing offerings differ from traditional penetration testing, because Bugcrowd uses the crowd to deliver the testing. For bug bounty, Bugcrowd offers two types: a continuous engagement that lasts for a year, and a time-based engagement. Bugcrowd also offers various integrations[30], such as Jira, Github, Trello, Slack, and an API for custom integrations. Finally, as mentioned in section 2.1.3, Bugcrowd uses their own vulnerability rating taxonomy for deciding the vulnerability severity.

HackerOne HackerOne was also founded in 2012[31] shortly after Bugcrowd. They currently offer three products[32]: Vulnerability Disclosure, Bug Bounty, and Penetration Testing. Similar to Bugcrowd, the penetration testing offering is delivered via the crowd, while still supporting compliance checks. HackerOne's bug bounty offering includes private programs, public programs, time-bound and virtual/live hacking events. Apart from their API, HackerOne also a range of integrations[33] including Jira, Bugzilla, Github, GitLab, Slack, Zendesk etc. As is evident, there are many similarities between HackerOne and Bugcrowd. The only difference from the comparison in this thesis is the severity calculation for which HackerOne[34] uses CVSS instead of implementing their own version (as did Bugcrowd).

Cobalt Cobalt was founded in 2013[35] and is heavily focused on penetration testing. Their products[36] include a number of penetration testing variants such as: web application pentest, mobile application pentest, external network pentest etc. As in the other platforms, the penetration testing is performed by a crowd member. However, contrary to Bugcrowd and HackerOne where anyone can sign up, Cobalt requires to have an invitation. Additionally, Cobalt performs heavy vetting of its members and has an initial probation period before allowing full membership.

[27] Synack Coverage Analytics, https://www.synack.com/wp-content/uploads/2017/11/Synack-Coverage-Analytics-11-2017-WEB.pdf, last visited on 05/24/2021
[28] Bugcrowd LinkedIn Page, https://www.linkedin.com/company/bugcrowd/about, last visited on 05/24/2021
[29] Bugcrowd Solutions, https://www.bugcrowd.com/solutions, last visited on 05/24/2021
[30] Bugcrowd Integrations, https://www.bugcrowd.com/products/platform/integrations, last visited on 05/24/2021
[31] HackerOne LinkedIn Page, https://www.linkedin.com/company/hackerone/about, last visited on 05/24/2021
[32] HackerOne offerings, https://www.hackerone.com/product/overview, last visited on 05/24/2021
[33] HackerOne Integrations, https://docs.hackerone.com/programs.html, last visited on 05/24/2021
[34] HackerOne Severity, https://docs.hackerone.com/programs/severity.html, last visited on 05/24/2021
[35] Cobalt LinkedIn Page, https://www.linkedin.com/company/cobalt_io/about, last visited on 05/24/2021
[36] Cobalt products, https://cobalt.io/services/pentest-service, last visited on 05/24/2021

Synack Founded in 2013[37], Synack offers a wide range of services. In addition to bug bounty and crowd-delivered penetration testing services, Synack also offers security testing through an AI-powered scanning tool[38]. Similar to Cobalt, Synack restricts crowd membership. To become a Synack member, whitehat hackers have to apply and provide personal details. A skills assessment is also performed prior to allowing membership.

Intigriti Intigriti is a european based bug bounty platform founded in 2016[39]. Currently, Intigritis only offering appears to be bug bounty. They do not have restrictions for becoming a member of the crowd, but bug bounty operators can set up restrictions on who can participate in their programs.

Yeswehack Yeswehack is another european based bug bounty platform, founded in 2013[40]. Similar to Intigriti, the only offering of Yeswehack is bug bounty. The different bug bounty program types supported are: private program, public program, and on-site events. Yeswehack advertises full compliance with the european general data protection regulation(GDPR), and holds certifications such as ISO 27001.

Safehats Founded in 2012[41], Safehats is an indian based bug bounty platform. Their current list of offerings[42] include enterprise bug bounty, a bug bounty hackathon, a startup-focused bug bounty program called „RISE“, and a community driven yellow pages for bug bounty programs.

As is evident, Bugcrowd and HackerOne are the largest platforms with the highest number of programmes. The other platforms are either focused on penetration testing (Synack & Cobalt), or bug bounty (Intigriti & Yeswehack). Table 4.2 shows the platforms and different offerings.

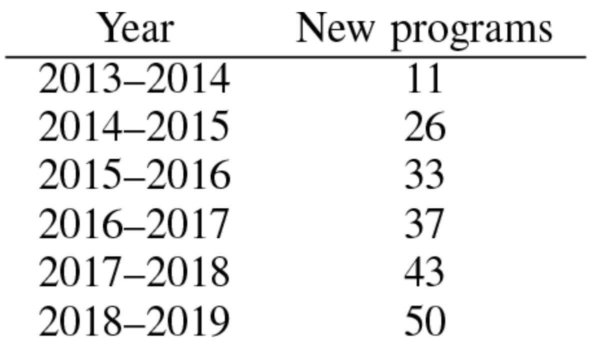
Platform	Offerings	Membership	Integrations
Bugcrowd	Bug Bounty Penetration Testing Vulnerability Disclosure Attack Surface Management	open	yes
HackerOne	Bug Bounty Penetration Testing Vulnerability Disclosure	open	yes
Cobalt	Penetration Testing	invitation-based	no
Synack	Bug Bounty Penetration Testing AI-powered scanning	application-based	no
Intigriti, Yeswehack, Safehats	Bug Bounty	open	no

Table 4.2: Comparison of bug bounty platforms

[37] Synack LinkedIn Page, https://www.linkedin.com/company/synack-inc-/about, last visited on 05/24/2021
[38] Synack products, https://www.synack.com/products, last visited on 05/24/2021
[39] Intigriti LinkedIn Page, https://www.linkedin.com/company/intigriti/about, last visited on 05/24/2021
[40] Yeswehack LinkedIn Page, https://www.linkedin.com/company/yes-we-hack/about, last visited on 05/24/2021
[41] Safehats LinkedIn Page, https://www.linkedin.com/company/safehats/about, last visited on 05/24/2021
[42] Safehats website, https://safehats.com, last visited on 05/24/2021

4.4 Current State of Adoption

As mentioned in the problem statement of this thesis, bug bounties have seen increasing popularity over the years. This has resulted in an increase of adoption by organizations as well as an increase in hacker membership. As shown by Walshe and Simpson[69], the number of launched programs on the HackerOne platform has been trending increasingly since 2013. The depiction of this trend is shown in figure 4.3.



Year	New programs
2013–2014	11
2014–2015	26
2015–2016	33
2016–2017	37
2017–2018	43
2018–2019	50

Figure 4.3: Trend of programs launched on HackerOne[69]

According to HackerOne's 2019 report[9], they serve more than 1,400 client organizations from 59 countries. This is an increase of 30% from the previous year. The United States remains the country with the most organizations that operate a running bug bounty, but HackerOne also reports clients from diverse countries such as Ghana, Slovakia, Aruba and Ecuador. In fact, the region with the biggest year-over-year increase in launched programs is observed in Latin America with 41%, followed by North America (34%), EMEA(32%), and finally APAC(30%). In a similar report, Bugcrowd[70] shows an increase of 42% in launched programs across all regions.

Looking at the industry perspective, HackerOne[9] reports that technology companies are the leaders in adoption and account for 60% of all active bug bounty programs. They are followed by internet & online services with 28%, and computer software with 21%. In terms of trends, HackerOne[9] shows the biggest year-over-year increase in federal government with 214%, followed by automotive (113%), telecommunications (91%), consumer goods (64%), and cryptocurrency (64%). Bugcrowd[70] on the other hand reports their biggest increase in the financial services industry with 71%, followed by an increase in retail with 50%, healthcare with 41%, technology with 26%, and a 13% increase in automotive industries.

Similar to the adoption of bug bounty by countries and industries, the hacker community has also seen widespread growth. According to Bugcrowd's „Inside the mind of a hacker" report[71], their community of whitehat hackers covers 6 continents and over 100 different countries. The majority of them is located in India, followed by the United States. Similarly, HackerOne[79] reports that 18% of their vulnerability submissions came from India, followed by United States with 11%. They also report hacker members from countries such as Panama, New Zealand, Hungary, Senegal etc. The popularity of bug bounties combined with easy participation and monetary rewards can be considered the main driver for emerging hackers around the globe.

Despite the popularity and demonstrated effectiveness of crowdsourced offensive security, HackerOne[9] reports that only 7% of the world's top companies have adopted this measure. Therefore, this work aims to analyze the problem space of integrating bug bounties into enterprise grade security programs. Furthermore, a case study is used to show that bug bounties can be effectively used to improve enterprise IT Security.

4.5 Comparison Against Other Methods of Security Testing

In section 2.2, the details of penetration testing and red teaming as offensive security measures are discussed. With this information at hand, a comparison of bug bounties against these security testing techniques can be provided. As shown by Maillart et al.[6], running a bug bounty program can seriously increase the number of vulnerability findings, but it comes with its own set of challenges. As such, bug bounties bring benefits and drawbacks into the offensive security toolkit of an organization.

The benefits of bug bounties are manyfold. The most important advantage and a key difference against other security testing methods is the large number of testers. As shown by HackerOne's annual report[79], they have a pool 600,000 security researchers which may be potential testers. Typically, the *active* number of testers is lower than advertised, and the actual number of testers is even lower, since this number is spread across all bug bounty programs operated by HackerOne. Nevertheless, the number of active testers is generally higher than the number of testers typically allocated in a penetration test..

According to Maillart et al.[6], bug bounties benefit from the engagement of a large crowd of testers because each security researcher offers a unique set of skills. The different backgrounds of security researchers almost always ensure the participation of an expert in the technology used by the target. As such, vulnerabilities which are typically missed by penetration testers can be discovered by bug bounty testers. The same argument is given by Magazinius et al.[7], who considers that a large and diverse bug hunter community is able to discover more vulnerabilities, simply by outnumbering the internal and external security teams of an organization or penetration testing services provider.

In addition to the benefit in number of testers, bug bounties also provide a better return of investment. Due to the nature of bug bounties, which only reward valid vulnerabilities found by the first security researcher, the cost (in form of monetary rewards) is directly correlated to the number of vulnerabilities found. This is different compared to penetration testing and red teaming where payment is contractually agreed upon, regardless of findings[72]. However, the pay-for-results model could become a drawback in cases where the target has many vulnerabilities and has had no prior testing. In this situation, the amount of vulnerabilities can quickly outgrow the cost of a penetration test which could potentially have identified the same vulnerabilities, or at least raised concerns about the security in general. Nevertheless, bug bounties remain a cost-effective offensive security testing measure for most applications, and by advertising the rewards for vulnerabilities upfront, they are able to maintain flexibility in controlling spend.

Finally, another benefit of bug bounties is testing times. Whereas other offensive security methods provide a picture of the security posture for a single point in time, bug bounties can run for any duration and even indefinitely. This delivers continuous security testing against the target application, and provides huge benefits to fast-paced organizations that deploy new functionality quickly. Because the targets are tested all the time, a newly introduced vulnerability can be quickly and cost-effectively identified, instead of waiting for the next penetration test cycle.

Just like bug bounties bring a lot of advantages in offensive security, there are challenges for achieving success with them. As shown by Al-Banna et al.[8], one of the major fears of organizations is the distrust in the security testers. In penetration testing and red teaming, the testers are employed by the security services provider and trusted individuals. Additionally, these testers are bound to the contract and rules of engagement for the penetration test. This is different in bug bounties, where potentially anyone can participate if the bug bounty program is open to the public.

There are no assurances about the location, tools and skills provided by the whitehat hacker. Although bug bounties have rules of engagement, there is no guarantee that bug hunters will adhere to them, and using legal means to enforce them is challenging because the majority of testers may live outside the country[79].

The quality of vulnerability reports received in bug bounties can also present a challenge, as shown by Al-Banna et al.[8]. Because participation is open to anyone, the quality of testing and reporting can differ. Furthermore, monetary incentives and the principle of rewarding only the first valid vulnerability report, motivates security researchers to rush sending the report without taking the proper time to verify their findings or provide a good summary. Researchers might also use an incorrect or misleading vulnerability severity. This can happen due to not having the business impact of the organization in mind, or trying to drive a higher reward. Finally, low quality reports can lead to longer processing times and a higher cost of maintaining the bug bounty program.

Other challenges with bug bounties include coverage, managerial experience and internal testing. Since bug bounty testers do not use a defined methodology, parts of the application might be missed or not tested at all. This is different in penetration testing, where a specific methodology ensures coverage of the entire target application. There is also the lack of expertise in bug bounties, especially for organizations looking to get started with it. Since this is a relatively new field, knowledge and capacity needs to be build within the organization. Penetration testing and red teaming on the other hand, have been around for much longer. As such, standards, regulation and industry have driven experience in organizations. Finally, bug bounties may not be able to test internal networks, as testing is typically performed remotely. In comparison, security providers often send penetration testers on site to test targets which may not be reachable from the internet.

Table 4.3 shows a comparison of bug bounties with penetration testing and red teaming. As is evident, each has their own advantages and disadvantages. Many of the challenges with bug bounties are already being addressed in current research, as shown by Malladi and Subramanian[14], Al-Banna et al.[8] etc. Nevertheless, adoption of bug bounties remains low as shown in section 4.3, and more practical research is needed to increase usage of bug bounties in enterprise organizations.

Characteristic	**Penetration Testing**	**Red Teaming**	**Bug Bounty**
Number of testers	few	few	many
Quality of reports	controlled	controlled	uncontrolled
Management overhead	low	low	high
Cost type	fix	fix	per vulnerability
Testing duration	1-2 weeks	1 week - 6 months	indefinite
Coverage	full	full	partial
Internal testing	yes	yes	no
Liability	yes	yes	difficult to enforce

Table 4.3: Comparison of bug bounty against other methods of security testing

4.6 Complementing Established Offensive Security Techniques With Bug Bounties

After having gained an understanding on how bug bounties work, and the difference between bug bounties and other offensive security methods, the question arises whether one can replace or complement the other. This is also part of this thesis' research questions as seen in section 1.4.

Bell et al.[72] argue that bug bounties cannot replace penetration tests, however, they can make penetration testing more effective. Besides the benefits of crowdsourced offensive security as mentioned in the previous sections, Bell et al. argue that penetration testing teams will have to work much harder when the target has an active bug bounty program. Typically, penetration testing providers want to have some results to show for their engagement. If their findings and report have little material, the chances of being asked for more penetration tests in the future diminish. It is not uncommon for penetration testing reports to include a number of low impact issues, just for the sake of reporting[72] . This results in a bad return of investment for the organization, as the cost for the penetration test does not change. However, when the target is actively being tested as part of a bug bounty program, many of these low impact issues and other vulnerabilities will be found and reported by the crowd. This leaves no easy findings for the penetration testing team to report. As a result, the penetration testing services company will be forced to utilize more senior staff and find harder vulnerabilities, to make sure they are providing value to the organization by reporting them. In this way, bug bounties not only provide continual assessment and discovery of vulnerabilities, but they also drive better value for other offensive security engagements.

A similar argument is given by Laszka et al.[75], who agree that whitehat hacker's efforts effectively complement traditional testing such as web vulnerability scanning. They also argue that it would be prohibitively expensive to directly employ a large and diverse number of testers, as they appear with bug bounties. This is mainly possible because of the pay-for-results model as described in section 4.1.3. Such statements are also observed from various industry experts[43]. They argue that bug bounty is a much needed alternative and provides benefits such as in-depth testing, something which is not common in penetration testing due to the shortage of security professionals. However, it is emphasized that bug bounties will not replace penetration tests. Reasons for that include coverage as discussed in section 4.5, and other vectors such as physical or social attacks. Bug bounties typically list these types of attacks as out of scope, and focus on technical vectors. As such, bug bounties have their own place in offensive security testing, next to penetration testing and red teaming.

Finally, the answer to research question 3 of this thesis can be provided. From the previous paragraphs and sections, it can be concluded that crowdsourced offensive security is able to improve and complement other offensive security methods. Bug bounties bring various advantages to security testing, and some of its disadvantages are remediated by traditional offensive security testing. As a result, bug bounties increase the effectiveness of penetration testing, and rightfully take their place in an organization's offensive security testing methods.

[43] How to use bug bounties with penetration testing to bolster your app sec, https://techbeacon.com/security/how-use-bug-bounties-penetration-testing-bolster-your-app-sec, last visited on 05/24/2021

5 Integration of Crowdsourced Offensive Security Into Enterprise IT

In the previous chapter bug bounties have been introduced and it was shown that they effectively complement other offensive security methods. The lack of adoption amongst enterprise organizations was also covered. The following chapter will show possibilities for integrating crowdsourced offensive security into enterprise IT. As discussed in section 4.1.2, there are two main possibilities for running a bug bounty program: an institutional approach where the bug bounty program is managed in house, and using a bug bounty services provider.

When using the institutional approach, an organization must provide the rules of engagement for the bug bounty program. These are also known as a vulnerability disclosure policy. The rules of engagement specify various aspects of the crowdsourced offensive security efforts. These aspects include: how to report a vulnerability, which targets should be tested, what type of testing is allowed, which vulnerabilities are eligible for a reward etc. The ISO 29147 standard[65] will be referenced and analyzed in more detail during this chapter, as it is the only standard concerned with crowdsourced offensive security in the context of this thesis.

The alternative to an institutional approach is using one of the available bug bounty service providers. A quick introduction of these platforms was given in section 4.3. In this chapter, guidelines for using bug bounty platforms by an enterprise organization will be provided. There are various pros and cons when using a bug bounty platform in enterprise organizations, which will be discussed in more detail.

By laying out the two possible approaches for integrating crowdsourced offensive security into enterprise IT, and providing guidelines for it, this chapter will give an answer to the first research question on possibilities for integrating crowdsourced offensive security.

5.1 Towards an Institutional Bug Bounty Program

With an institutional approach, the organization decides to take management of their bug bounty program in their own hands. This requires the creation of rules of engagement, creating operational processes and assembling a team to manage the program. Bell et al.[72] argue that running a bug bounty program in house not only gives ultimate control over it, but may also be cheaper if a technical security team is already established.

As shown in section 4.1.3, typical items covered in the rules of engagement are the targets to be tested, various restrictions and legal requirements, information about rewards and communication requirements etc. While real attackers do not follow any rules, the concept of rules of engagement is well established in the bug bounty community and provides a basis for participation[73].

A team for handling the daily operations needs to be established when running an institutional bug bounty. This includes not only technical security personnel for vulnerability verification, but also engineers that will work to fix them, support personnel and a leader to manage them. There is also the stakeholder aspect of the program, which defines various people as stakeholders within the organization.

Finally, this section will provide an example rules of engagement, which can as a template or starting point for organizations to create their own version.

5.1.1 Creating the Rules of Engagement

As shown in section 4.1.3, the rules of engagement are a necessary foundation for running a bug bounty program. Weulen Kranenbarg et al.[80] argue that it also serves as an invitation for white-hat hackers to report any vulnerabilities they find in an organization's assets. Generally, the rules of engagement are the only information source for crowdsourced security testers about a particular bug bounty program. Hence, it must include all details about the bug bounty. A bug bounty program with more specific rules of engagement tends to be more successful, as shown by Laszka et al.[73] in their study of rules of engagement.

The ISO 29147 standard[65] suggests that at a minimum, a vulnerability disclosure policy should include the following: how the vendor would like to be contacted, secure communication options, setting communication expectations, information that is useful in a vulnerability report, out of scope services, and how reports are tracked. The standard also defines optional items such as crediting the finder, coordinated public disclosure and distribution of vulnerability reports. Weulen Kranenbarg et al.[80] further suggest that a vulnerability disclosure policy should include verbiage to prevent unauthorized public disclosure or passing the vulnerability details to third parties. As mentioned in section 2.1.3, part of the security community argues that disclosing vulnerabilities publicly is the only reliable way to pressure organizations in improving their security[1]. To set correct expectations around this behaviour, the rules of engagements must clarify if and when public disclosure is allowed, and how the process is coordinated.

Having these details in the rules of engagement can be beneficial to both parties involved, as it provides certainty. For the organization, the policy provides the benefits of learning about vulnerabilities and potential risks to their systems. For the whitehat hacker reporting the vulnerability, it provides two main benefits: first, by acting based on the organization's vulnerability disclosure policy, the whitehat hacker is not acting illegally and is ensured no criminal investigation. Second, by reporting a vulnerability as a part of a disclosure policy, the whitehat hacker may receive some form of reward as discussed in section 4.1.3.

For enterprise organizations, this work suggests the following elements to be included in the rules of engagement:

1. Introduction
2. Reporting a Vulnerability
3. Scope
4. Excluded Vulnerability Types
5. Testing Restrictions

[1] Schneier: Full Disclosure of Security Vulnerabilities a 'Damned Good Idea, https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html, last visited on 05/24/2021

6. Geolocation Restrictions
7. Handling Vulnerability Reports on Third-Party Software
8. Hall of Fame
9. Public disclosure

1. Introduction

It is recommended to start the rules of engagement with an introduction paragraph, introducing the bug bounty program and organization. In this paragraph, an organization can describe what they plan to achieve with the program, and state their commitments towards crowdsourced security researchers. This paragraph can also be used to introduce what the organization does, and why it choses to engage crowdsourced offensive security.

The introduction segment is helpful in many ways. It shows whitehat hackers that an organization cares about security in general, and also that it is committed to work with them. It also explains what the rules of engagement are (and what they are not), for new crowdsourced security testers, non-security professionals and the general public interested in it.

2. Reporting a Vulnerability

This section of the rules of engagement provides guidelines on how to report a vulnerability. There are two aspects of this process:

- Communication options for reporting a vulnerability.
- Information to be included in a vulnerability report.

Communication options for reporting a vulnerability. Communication is one of the most important details of a vulnerability disclosure policy. As shown in section 3.3.2, a vulnerability report as part of a bug bounty includes all necessary details to reproduce the issue. Because those details could allow a malicious actor to abuse a vulnerability, it is important to ensure secure communication. In addition to this, a vulnerability disclosure policy must also specify the available means of communication.

As specified in the ISO 29147 standard[65], part of the minimum requirements for a vulnerability disclosure policy is to specify how the vendor would like to be contacted. Typical options for reporting a vulnerability are e-mail or completing a web form. Each has its advantages and disadvantages which will be covered in the following sections.

For communication via email, one or multiple email addresses should be provided. The ISO 29147 standard[65] suggests to use an e-mail alias such as: security-alert@example.com, security@example.com, secure@example.com, psirt@example.com, or csirt@example.com. Reporting vulnerabilities via email is a common method used in an institutional bug bounty approach. For example, Deutsche Telekom[2] requires vulnerability reports to be send to *bugbounty@t-mobile.cz*. One advantage for using e-mail as the communication option is the ease of use and IT infrastructure which is generally available in enterprise organizations. This makes e-mail a preferred choice

[2] Closing security gaps, https://www.telekom.com/en/corporate-responsibility/data-protection-data-security/security/details/closing-security-gaps-360054, last visited on 05/24/2021

for quick setup.

In case the desired communication is via web form, a link to the submission form should be provided. Web forms have the advantage of allowing more control over the structure of the vulnerability information being reported. As mentioned previously, the lack of structure in vulnerability reports sent via emails is a disadvantage when compared to a web form. Since bug bounty participation may be open to anyone, this can result in a lot of noise or vulnerability reports with incomplete information. When using a web form, separate fields can be used for the various pieces of information required in a vulnerability report. Some (or all) of these fields can be made mandatory, further enforcing structural requirement on information reported as part of a vulnerability report.

Another form of communication that has become popular in recent years[3] are platforms such as Slack[4] and Discord[5]. Besides real-time messaging between users, these platforms also offer channels (essentially group conversations based on topics) for collaboration, file uploads, various integrations with other tools etc. Therefore, it is an option that can be considered for reporting and receiving security vulnerabilities. Compared to communication via email, this form offers a less formal means of communication between security researchers and bug bounty operators. The real-time aspect allows for a more fluid stream of messages, and potential disputes may be resolved faster. This form of communication can only be effective if participants are present, therefore, issues such as staffing the platform with security engineers must be considered. Finally, some security engineers may prefer email, which should always be present as an option.

As mentioned before, vulnerability reports contain sensitive information which must be protected against eavesdropping. As such, the ISO 29147 standard[65] cites that depending on the selected communication form, a secure medium of transmission should be provided. For email, technologies such as PGP or S/MIME can be used, as shown by Garfinkel et al.[81]. Likewise, if communication is specified via web form, it can be served over a secure connection (HTTPS).

Information to be included in a vulnerability report. Besides communication options, a vulnerability disclosure policy should specify the pieces of information required in vulnerability reports[65]. This information contains details about the vulnerability type, security impact, reproduction steps etc.

Bugcrowd provides some guidelines[6] for useful information to be sent in a vulnerability report. They suggest that the following information should be provided in a vulnerability report as part of a bug bounty:

- Summary Title
- Technical Severity
- Vulnerability Details
  - Bug URL

[3] What Is Slack, and Why Do People Love It?, https://www.howtogeek.com/428046/what-is-slack-and-why-do-people-love-it, last visited on 05/24/2021
[4] Slack is where work happens, https://slack.com/intl/en-at, last visited on 05/24/2021
[5] Your place to talk, https://discord.com, last visited on 05/24/2021
[6] Submitting Vulnerability Using Embedded Form, https://docs.bugcrowd.com/researchers/reporting-managing-submissions/reporting-a-bug, last visited on 05/24/2021

  - Description
  - HTTP Trace Dump
  - Additional Information
  - Attachments

- Email Address

- Confirmation of Agreement to Terms & Conditions

Summary Title This field is the title of the vulnerability report. It usually includes some information about the vulnerability type and the target that is affected.

Technical Severity This field allows the whitehat hacker to provide their assessment of severity. Note that this severity assessment is from a *technical* standpoint, and the organization running the bug bounty program will assess the final impact to the business. One of the rating systems shown in section 2.1.3 can be used here, for example CVSS.

Vulnerability Details These sub-fields include the details of the vulnerability itself. The bug url holds the URL of the target endpoint where the vulnerability has been identified. The description field provides space for the whitehat hacker to describe the vulnerability and provide information to reproduce for the organization. The HTTP trace dump is a field where whitehat hackers can provide the full HTTP request/response logs which trigger the vulnerability. A field for additional information gives the whtiehat hacker space to provide information such as tools used, the type of browser etc. Finally, the whitehat hacker is able to attach files such as screenshots and videos which are useful for the security team trying to reproduce the issue.

Email Address This field obtains an email address from the vulnerability reporter, for further communication and attribution. This field can be made optional if the organization permits anonymous reports.

Confirmation of Agreement to Terms & Conditions The last field is a checkbox that confirms the whitehat hacker has read and agrees to the terms & conditions of the bug bounty program. It is recommended to make this mandatory, which then serves as a reminder that the rules of engagement must be followed.

Of course this is not the only way to construct a web form for vulnerability reports. For example, Google[7] uses a form wherein a list of targets is given. The form also has fields for specifying whether the issue is publicly known or not. Organizations are free to build and modify the form to fit their needs, however, more information obtained during the reporting can results in less communication required afterwards for vulnerability verification.

3. Scope

The scope section in the rules of engagement defines the exact targets for which testing is permitted. This can be a single target, a list of targets, or the entire IT infrastructure of an organization. As shown by Erbes et al.[82], the IT infrastructure in enterprises is changing rapidly. The availability of cloud services, consumer devices and increasing cross-enterprise collaboration fundamentally changes the traditional system of in-house IT. This has also led to an exponential growth in the

[7] Report a security vulnerability, https://www.google.com/appserve/security-bugs/m2/new, last visited on 05/24/2021

size of IT assets owned and operated by enterprise organizations[82].

From a crowdsourced offensive security perspective, authorizing security testing on the entire infrastructure will generally lead to more vulnerability reports[12]. There are examples of large enterprises[8] who have successfully operated bug bounties with an open scope and issued rewards in total of millions of dollars. In this sense, an open scope means any asset owned by the organization is considered in scope. However, inviting the crowd to test any system owned by the organization may lead to undesired results and instability. As mentioned previously, nowadays enterprise IT infrastructures encompass a huge number of assets, some of which may not be suitable for crowdsourced security testing. For example, there may be legacy systems which may not withstand security testing with modern tools. Other assets may be under active development or waiting to undergo internal testing. If the scope definition is open and includes these assets, whitehat hackers may find them and report valid vulnerabilities which must be rewarded. Depending on the number of vulnerabilities, this may become exponentially more expensive than running a traditional penetration test.

In order to direct whitehat hackers to the desired targets for crowdsourced security testing, a vulnerability disclosure policy should clearly specify the assets that are in-scope. As shown by Laszka et al.[73], these statements define the exact scope of the bug bounty. This can include production targets, staging targets or assets set up specifically for security testing. Additionally, organizations may specify other components such as APIs, mobile and desktop applications and even physical products if available. In general, a detailed list of in-scope targets will lead to better information for researches and correlate to a lower number of false positive vulnerability reports[73].

There are a number of ways to define different types of targets such as: web applications, apis, mobile applications etc. Table 5.1 shows common examples of targets listed as in-scope and their interpretation.

Scope	**Meaning**
www.target.com	The target at www.target.com and all its paths are in scope
*.target.com	Any subdomain of target.com (including target.com itself) are in scope
api.target.com	The APIs at api.target.com are in scope
https://github.com/exampleinc/target	The application source code at the given link is in scope
Android app at https://play.google.com/...	The Android application found via the given link is in scope
iOS app at https://apps.apple.com/...	The iOS application found via the given link is in scope
Target SDK	The target software development kit is in scope

Table 5.1: Examples of in-scope targets for a vulnerability disclosure policy

For web targets, if a domain name is given, whitehat hackers will likely interpret that any path and functionality within that domain is in scope for testing. If a more restrictive definition is desired, the individual paths can be listed in the targets table. In cases where the domain is prepended by a

[8] Verizon Media, PayPal, Twitter Top Bug Bounty Rankings, https://threatpost.com/verizon-media-paypal-twitter-bug-bounty-rankings/157040, last visited on 05/24/2021

star (*), all subdomains of the target are considered in scope. The same interpretation of path and functionality applies for all subdomains in this case.

If the target is open source, asking whitehat hackers to set up their own test instance can be an effective approach with less overhead for the organization. This is also a favourable situations since the crowd can perform whitebox testing, and list the vulnerable code. For organizations, it is recommended to have an internal test instance for vulnerability verification.

The ISO 29147 standard[65] also recommends specifying „out of scope“ services. These are targets for which testing is not desired, but also include other security related events such as security incidents or third party services. Out of scope services may be listed in a similar fashion to in-scope services, by calling out explicitly that testing is not permitted. However, caution should be taken when specifying both in-scope and out-of-scope targets, as this may lead to confusion for targets that fall in neither category. In this situation, specific targets should be included in only one of them, while the other category provides a statement about „everything else“. For example, if in-scope targets have been provided, the following statement can be used to default all other assets as out-of-scope:

> *„Any domain/property not listed in the in-scope targets section is considered out of scope and should not be tested.“.*

While this may be common sense, in situations where targets are specified in the out of scope section, it should be clarified that only targets owned by the organization are considered in scope. The following statement can be used to declare this:

> *„Unless listed in the out-of-scope section of this policy, all assets owned by Acme Inc. are considered valid targets.“.*

Finally, the scope section must clarify the situation for vulnerability reports on third-party systems. Generally, active testing on third-party systems is not desired as those systems are not owned by the bug bounty operator. Hence, no permission for testing on these systems can be provided. Therefore, it should be clearly specified that testing third-party systems is out of scope. For situations with known vulnerabilities in third-parties or misconfigurations, it is recommended to handle them as defined in the section on handling vulnerabilities in third-party systems below.

There could also be situations where a new vulnerability becomes public and vulnerability reports from whitehat hackers immediately start to come in. While this is valuable information, it leaves no time for the organization to patch their systems. Therefore, it is recommended to set a patching-period delay for public vulnerabilities. For example, the vulnerability disclosure policy could state that reports for publicly disclosed vulnerabilities will not be accepted for the first 30 days after publication. This allows for ample time to patch systems, while retaining valuable information from reports after the patching period has expired, which may find systems missed during initial remediation.

4. Excluded Vulnerability Types

The rules of engagement also typically define a list of vulnerability types which are excluded from the bug bounty program. There are various reasons why an organization may decide to exclude some vulnerability types. Typically, many low impact vulnerabilities such as session-management related issues are excluded. There can also be exclusions for attacks that require social engineering or a high-privileged access to succeed. Bugcrowd[9] suggests that exclusions are essential to a successful bug bounty brief and includes low impact vulnerabilities, intended functionality that may appear vulnerable, and issues that are already known to the organization.

In their standard disclosure terms[10], Bugcrowd provides a list of common non-qualifying submissions types. These are vulnerability types that generally have a low impact and an organization does not want them to be reported through their bug bounty program. Table 5.2 shows Bugcrowd's common list, which is a good starting point for organizations to build their own version.

Excluded Vulnerability types
Descriptive error messages (e.g. Stack Traces, application or server errors)
HTTP 404 codes/pages or other HTTP non-200 codes/pages
Banner disclosure on common/public services
Disclosure of known public files or directories, (e.g. robots.txt)
Clickjacking and issues only exploitable through clickjacking
CSRF on forms that are available to anonymous users (e.g. the contact form)
Logout Cross-Site Request Forgery (logout CSRF)
Presence of application or web browser 'autocomplete' or 'save password' functionality
Lack of Secure and HTTPOnly cookie flags
Lack of Security Speedbump when leaving the site
Weak Captcha / Captcha Bypass
Username enumeration via Login Page error message
Username enumeration via Forgot Password error message
Login or Forgot Password page brute force and account lockout not enforced
OPTIONS / TRACE HTTP method enabled
SSL Attacks such as BEAST, BREACH, Renegotiation attack
SSL Forward secrecy not enabled
SSL Insecure cipher suites
The Anti-MIME-Sniffing header X-Content-Type-Options
Missing HTTP security headers

Table 5.2: List of common non-qualifying vulnerability types by Bugcrowd[11]

Another common exclusion is functionality that may appear vulnerable, but works as intended. For example, web applications that allow customization of their user interface may receive reports for content spoofing or content injection vulnerabilities. While this may appear to be a valid vulnerability, in applications designed to be customized, this can be functionality that is working as intended. To avoid confusion from whitehat hackers, such functionality should be explicitly called out as not being a vulnerability in the context of the application.

Finally, vulnerabilities which are already known to the organization (for example due to previous penetration tests) should be excluded if possible. Excluding such issues without giving away

[9] Essential to a Successful Bounty Brief: Exclusions, https://www.bugcrowd.com/blog/exclusions-essential-to-successful-bounty-brief/, last visited on 05/24/2021

[10] Bugcrowd Standard Disclosure Terms, https://www.bugcrowd.com/resource/standard-disclosure-terms, last visited on 05/24/2021

[11] Bugcrowd Common Non-qualifying Submission Types, https://www.bugcrowd.com/resource/standard-disclosure-terms, last visited on 05/24/2021

too much information about the vulnerabilities may be difficult, however, it is recommended to at least have a callout in the rules of engagement that set correct expectations with the crowd. If a target has a high number of known vulnerabilities that have yet to be fixed, it is better to perform a thorough remediation before engaging in crowdsourced security testing.

5. Testing Restrictions

Since a vulnerability disclosure policy is the main document for communicating the rules of engagement, it needs to also contain information about restrictions. These include actions that are not desired from the organization operating the bug bounty, and therefore, are prohibited. Disregarding these testing restrictions typically results in disqualification from receiving a reward for the reported vulnerability. However, violations may also result in a ban or even legal actions from the bug bounty operator against the reporter.

As shown by Laszka et al.[73], testing restrictions may include a number of categories such as: restrictions for automated scanning, social engineering, denial of service attacks, interaction with other user accounts etc.

Automated scanning Vulnerability scanners are not perfect and often report false positives, as shown by Makino and Klyuev[83]. Combined with the ability for anyone to participate in open participation bug bounties, there can be many false positive reports from inexperienced whitehat hackers. This increases the noise and is generally undesired. Additionally, vulnerability scanners themselves are noisy and inject lots of traffic into the target application. This may be undesired when the target is a production environment or not intended to handle high traffic loads. Hence, the permission for using automated scanning should be evaluated and specified in the vulnerability disclosure policy.

Social engineering Social engineering is the process of manipulating people into performing actions or divulging confidential information[84]. If this type of testing is desired, it is typically performed by trusted individuals as part of a penetration test or red-team engagement. Hence, this type of testing is generally excluded from crowdsourced security testing, because there is no direct relationship between the organization and the security tester.

Denial of service Denial of service attacks are described in detail in section 2.1.6. While it is useful information in knowing whether a target is vulnerable to denial of service attacks, it is typically excluded in crowdsourced security testing. This is especially important when in-scope targets are production applications, to avoid making them inaccessible to normal users while testing is under way. In special cases where dedicated targets for security testing are available, denial of service can be allowed as there is no impact if the target crashes.

Interaction with other user accounts Another typical undesired behaviour is for whitehat hackers to interact or access legitimate user accounts. This is another problem which commonly applies only to production systems. Because whitehat hackers are required to provide proof of concepts for their findings, they will not shy away from accessing legitimate user's data. This is generally undesired by organizations, as it would result in leakage of personal information of those users and considered an incident. Hence, this is a typical restriction as part of the rules of engagement.

Pivoting As mentioned in section 4.5, whitehat hackers will try to find the highest impact possible to derive a bigger reward. In this attempt, sometimes they may use a vulnerability on one target to reach another target. This is commonly called „pivoting"[46]. In general, pivoting is not a problem as long as all targets are in scope. This also provides a better picture of what a real attacker is possible to achieve. However, organizatons may also decide to completely disallow pivoting. For example, Facebook disallows exploitation of a vulnerability for any purpose other than testing its validity[12]. Instead, Facebook will work internally to determine the maximum impact an issue may have, and issues rewards according to that. Whether pivoting is allowed or prohibited, it should be clearly called out as part of the rules of engagement. Having no statement about pivoting may lead to different interpretations by crowdsourced security testers, and as a result, undesired behaviour of whitehat hackers.

As can be seen, there are various situations which arise during crowdsourced offensive security testing which must be clarified. The list above is not final and does not work for every organization. For example, there may be organizations that would like to see testing for denial of service and understand whether they can withstand such attacks. It should be noted that many situations may arise during testing that can be found undesired and added to these restrictions after launching the bug bounty program. The rules of engagement are not intended as a static resource, and adaptions should be made on the go. Ideally, whitehat hackers are notified about changes through email or other mechanisms. If this is not possible, it should be called out that rules of engagement may change and should be frequently revisited.

6. Geolocation Restrictions

Another important aspect that should be regulated in a vulnerability disclosure policy is restrictions on geolocations. Theoretically, a bug bounty can be open for participation to anyone in the world. However, bug bounties still have to abide by local and international law. There can also be restrictions imposed by organizations themselves on who can participate. Therefore, in practice bug bounties are often not open to everyone, since these restrictions prevent some testers from participating.

The first and most important restriction is related to nationality of the whitehat hackers. Depending on the country of origin where the bug bounty operator's headquarters are located, there may be restrictions or sanctions imposed on other countries. This is usually regulated by law and may cause legal issues if one does not comply. For example, the United States has imposed various political and economical sanctions on Iran, as shown by Dizaji and Farzanegan[85]. While they may be far from IT and bug bounties, laws still prohibit any type of economical trade with citizens from said country. Hence, a bug bounty operator will want to notify potential whitehat hackers from these countries in advance, by having such a clause in their rules of engagement.

A second but also important restriction typically imposed by bug bounty operators is to disqualify their own employees from participation. There are good reasons for doing this: to prevent conflict of interest. As mentioned previously, some organizations can spend millions in rewards which results in quite a lucrative income for bug bounty hunters. In fact, HackerOne reports that seven members of their crowd have surpassed $1 million dollars in rewards[13]. Therefore, in order to prevent employees from intentionally writing insecure software, they are not allowed to participate in the bug bounty program of their employer. Employees would also have an advantage against the larger crowd, profiting from internal information and access to internal systems.

[12] Facebook whitehat, https://www.facebook.com/whitehat, last visited on 05/24/2021

[13] Congratulations, Cosmin! The world's seventh million-dollar bug bounty hacker, https://www.hackerone.com/blog/congratulations-cosmin-worlds-seventh-million-dollar-bug-bounty-hacker, last visited on 05/24/2021

Finally, bug bounty operators may decide to restrict participation of whitehat hackers under the legal minimum age of employment. This is because monetary rewards are seen as payment against „work“ done by security researchers, and many countries specify a minimum age for legal employment[86].

A vulnerability disclosure policy typically also includes a legal clause. This is used to specify details about legal claims and issues. The most important callout on a legal clause is the commitment to not legally persecute whitehat hackers, which perform security testing as part of, and in compliance with the rules dictated in this vulnerability disclosure policy. The legal clause may also callout whitehat hackers to remember to abide by their local laws, as well as the laws of the country hosting the bug bounty operator. Finally, another legal aspect that is covered in this section, is the right to modify the rules of engagement at any time. Some organizations even reserve the right to not issue rewards. Of course this may shy away whitehat hackers, as they are not ensured payment for their work. As long as this is called out on the vulnerability disclosure policy, whitehat hackers have access to this information and can decide whether or not they would like to participate.

7. Handling Vulnerability Reports on Third-Party Software

As shown by Al-Banna et al.[8], organizations have increasingly complex systems which rely on many external services such as cloud services, external APIs and programming libraries, etc. Naturally, vulnerabilities in these services will be identified as part of running a bug bounty program. As shown in the scope section, typically all third-party systems are excluded from the bug bounty program by declaring them out of scope. However, this could lead to having the organization’s data stored on vulnerable (third-party) systems. For example, a third-party services provider may have already released a patch which needs to be installed. Or it may be that the third-party doesn’t have enough resources and takes a long time to patch the vulnerability, putting the security of the organization’s data into their hands.

For organizations running a bug bounty program, an internal process should be defined that handles vulnerability reports on third-party systems. For vulnerabilities where remediation can be performed by the organization itself (for example, by applying a patch), it can be handled via the standard process as defined in section 6.4. If the vulnerability must be patched by the third-party, a process for communicating and tracking the issue should be specified. Finally, if the vulnerable third-party is not able to remediate the issue, alternative options for removing the risk should be identified.

8. Hall of Fame

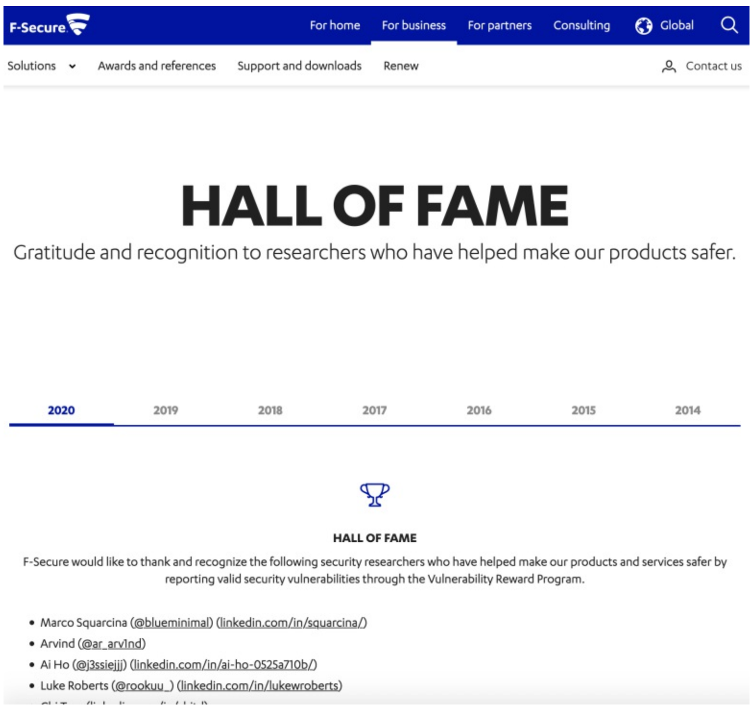
As shown by Bell et al.[72], crediting the whitehat hacker for having identified a valid vulnerability is a very important aspect of a bug bounty program. Credits are typically listed in a „Hall of Fame“, which is a publicly accessible resource. The format for crediting a whitehat hacker can be just the name of the finder, a twitter handle, a link to a website or more. Some organizations also list the date of the finding, and the vulnerability type[14].

Malladi and Subramanian[14] argue that a hall of fame can act as a motivator for whitehat hacker participation. By listing new comers but also top contributors, an organization’s bug bounty program becomes attractive for testers motivated by fame and reputation. Some organizations even

[14] Deutsche Telekom Acknowledgements, https://www.telekom.com/en/corporate-responsibility/data-protection-data-security/security/details/acknowledgements-358300, last visited on 05/24/2021

gamify their hall of fame and have special leaderboards, such as Github[15]. Crediting researchers and managing reputation is also a common part of bug bounty platforms, which use this information for inviting testers to private bug bounty programs[16].

Picture 5.1 shows the hall of fame for F-Secure's bug bounty program[17].



HALL OF FAME

Gratitude and recognition to researchers who have helped make our products safer.

HALL OF FAME

F-Secure would like to thank and recognize the following security researchers who have helped make our products and services safer by reporting valid security vulnerabilities through the Vulnerability Reward Program.

- Marco Squarcina (@blueminimal) (linkedin.com/in/squarcina/)
- Arvind (@ar_arv1nd)
- Ai Ho (@j3ssiejjj) (linkedin.com/in/ai-ho-0525a710b/)
- Luke Roberts (@rookuu_) (linkedin.com/in/lukewroberts)

Figure 5.1: F-Secure Hall of Fame

9. Public Disclosure

As mentioned in section 2.1.3, public disclosure of vulnerabilities is a hot debate in the cybersecurity community. Whereas one side argues that public disclosure of vulnerabilities helps to improve the security of software, the other side argues that by making such information public, it benefits attackers and results in higher risk of exploitation[35].

By default, it is recommended that public disclosure of vulnerabilities as part of a bug bounty is not allowed. Instead, coordinated vulnerability disclosure should be used. As shown by Weulen Kranenbarg et al.[80], coordinated vulnerability disclosure is a well-known practice for finding and patching flaws in IT-systems. Using this practice, a vulnerability reported by a whitehat hacker will only be disclosed *after* it has been patched.

Laszka et al.[73] argue that organizations are generally not in favour of public disclosure. By publicly disclosing the vulnerabilities reported to them, organizations are concerned about the perceived image of their internal systems and application security. In their study of 111 rules of

[15] GitHub Security Bug Bounty, https://bounty.github.com, last visited on 05/24/2021
[16] Reputation, Signal & Impact Calculation Enhancements, https://www.hackerone.com/blog/reputation-signal-impact-enhancements-whats-changing-and-why-it-matters, last visited on 05/24/2021
[17] Hall of Fame, https://www.f-secure.com/en/business/programs/vulnerability-reward-program/hall-of-fame, last visited on 05/24/2021

engagement for bug bounty programs, only 38 programs had statements about public disclosure in their vulnerability disclosure policy. However, there may be cases where public disclosure is desired and beneficial. For example, immediately disclosing a vulnerability in open source software could speed up remediation through community contributions.

Regardless of whether public disclosure is allowed or not, it should be specified in a vulnerability disclosure policy to set correct expectations with the researchers. It is recommended to have a statement related to coordinated disclosure, with an emphasis on a case by case analysis. As shown by Ruohonen et al.[11], only relatively few whitehat hackers like to disclose their vulnerability reports. This can be traced back to the desire of keeping methods and tools secret, for the competitive advantage against other whitehat hackers. With this mind, the expected number of requests for public disclosure of vulnerability reports should be low. As such, a case by case analysis would be feasible. Such an approach also gives the organization time to analyze and think of possible outcomes of disclosing a vulnerability report. There are situations where a public disclosure may be beneficial and show that the organization cares about security.

Finally, the rules of engagement should provide a statement to clarify consequences of breaching public disclosure requirements. Typically, breaching any part of the rules of engagement results in (at least) disqualification from receiving a bounty. If the breach is severe, disqualification from further participation in the bug bounty program may also be issued. To this end, it is important to clearly specify what constitutes a breach of public disclosure requirements, and the consequences of doing so.

5.1.2 Assembling the Operations Team

With the rules of engagement defined, an organization needs to assemble the team for managing the bug bounty program. It is important to have enough resources for managing the daily operations. As shown by Bell et al.[72], a timely response to vulnerability reports is a key success factor for running a bug bounty program, therefore, these response capabilities need to be established. There are also other activities that require staffing dedicated personnel who is committed for successfully running a bug bounty program.

As shown in section 4.1.3, there are various activities during a bug bounty program lifecycle. The main responsibility and majority of efforts are with the security team that handles vulnerability reports. This is usually a team of security engineers that work to reproduce and validate the vulnerabilities reported by whitehat hackers. Typically, they also handle communication with security researchers, to let them know about the status of their reports. Although it is recommended to have a team of security engineers responding to vulnerability reports, it can be staffed with a single person. One key aspect is to maintain a timely response to vulnerability reports and not let crowdsourced security testers wait for days. If response times are long, whitehat hackers may become discouraged from further testing. Another critical aspect is the technical competence of the security team. While many whitehat hackers will report low or trivial issues, there are also highly skilled testers that will report very technical vulnerabilities. If the security team is unable to understand the report and continues to push the whitehat hacker for additional information, it may appear that the organization is not investing in their team's security skillset.

Other typical members of the organization that have responsibilities and (indirectly) affect the success of the bug bounty program are product managers and engineers. A product manager takes reported vulnerabilities which have been verified from the security team, and schedules patches for them. Software engineers will then work to patch the vulnerabilities and notify the security team of the remediation. It is important that both product managers and engineers are involved in the bug bounty program. They should at least understand what the program is, what purpose it has and when their involvement is required. This helps to have a smooth transition from vulnerability report to a ticket in the bug tracking system of the enterprise organization.

As can be seen, multiple organization members have different roles and require a tight communication loop. Therefore, it is recommended to have a role which coordinates all members and ensures the success of the bug bounty program. This can be accomplished by creating a „bug bounty coordinator“ role, or using a generic project manager. The bug bounty coordinator would ensure the contribution of individual members, the communication flow, and general health of the bug bounty program. Additionally, as shown by Zhao et al.[15], whitehat hackers need incentives to keep testing for security vulnerabilities. This could be handled by the bug bounty coordinator, or it could be separated into its own role. Finally, a member of the organization needs to handle support-related issues, when whitehat hackers have problems with accessing infrastructure or similar issues.

Finally, the process for issuing rewards needs to be staffed. This includes activities such as: deciding the reward type, determining the amount, delivering the reward, and updating the hall of fame (if available). If monetary rewards are awarded, it is recommended to create a reward panel. Since determining the amount of the reward requires technical as well as business knowledge, a panel which consists of members from different stakeholder departments will be best positioned to make such a decision. A good example of reward decision management via a panel is Google's Vulnerability Reward Program[18]. Once the amount has been decided, the reward must be delivered. Typically, this responsibility is left to the finance department, and the bug bounty coordinator role manages the process. The bug bounty coordinator is also responsible for updating and maintaining the „hall of fame“ list, which credits researchers who have found a valid vulnerability as part of the bug bounty program.

5.1.3 Specifying Stakeholders and Accountability

Determining the stakeholders and accountable personnel within an organization is another aspect required for running a bug bounty program. Stakeholders are people who have an interest in the success of the program, and may be accountable for it. Obvious stakeholders of crowdsourced offensive security are the CISO of the organization, as well as the bug bounty coordinator. These roles work closely together to achieve the goals and purpose of the bug bounty program.

Besides the previously mentioned organization members that are involved in running and managing a bug bounty program, other stakeholders may include departments such as legal, public relations and finance. This is because bug bounty management requires interaction with external entities, and from their perspective all interactions are observed on behalf of the company as a whole. Involving the legal department is especially important, as they handle the liability parts in case a whitehat hacker breaks the rules of engagement and accesses sensitive information. The legal clause section of the vulnerability disclosure policy is typically handled by the legal department. They should also be consulted about restrictions as described in section 5.1.1. The other non-obvious stakeholder is public relations or the marketing team. As shown by Bell et al.[72], the inherently public nature of bug bounty programs can result in positive as well as negative PR, and it mostly depends on the testers's interaction experience with the bug bounty program[72]. For example, a popular videoconferencing platform called „Zoom“ received a lot of negative media

[18] Google Vulnerability Reward Program (VRP) Rules, https://www.google.com/about/appsecurity/reward-program, last visited on 05/24/2021

attention after a researcher publicly disclosed a critical vulnerability[19]. However, simply having a vulnerability disclosure policy can result in good publicity[80]. It shows that an organization cares about its security, but also about security researchers that are trying to help them improve it. The existence of the ISO 29147 standard[65] and the fact that even the Pentagon[87] successfully ran a bug bounty program are good indicators of the positive aspect of bug bounty programs. Finally, the finance department is considered a stakeholder since they secure the budget and are responsible for delivering monetary rewards. As mentioned previously, the logistics of delivering the money to various countries in the world may be tricky, and requires close cooperation with financial experts.

This thesis suggests that members from following departments should be considered when determining the stakeholders of an organization's bug bounty program:

1. Chief Information Security Officer
2. Product Manager
3. Engineering
4. Bug Bounty Coordinator
5. Finance
6. Marketing
7. Legal

5.1.4 Rules of Engagement Example

In this section, an example for rules of engagement is provided. It acts as a template that may be used as a basis for building their own version. As discussed in section 5.1, a vulnerability disclosure policy invites whitehat hackers to report vulnerabilities they find to the organization. For this process to work, rules must be clearly stated and advertised. In the following paragraphs, sample content for each part of the rules of engagement as described in section 5.1 will be provided.

Introduction An introduction paragraph explains what the policy is for:

> *Acme Inc. is committed to resolving vulnerabilities to meet the needs of its customers and the broader technology community. This document describes Acme Inc.'s policy for receiving reports related to potential security vulnerabilities in its products and services.*

Reporting a vulnerability This section lists the communication options and process as defined in section 5.1.4. Text colored in blue represents links to URLs and should be adjusted by the organization as needed:

> *If you have identified a vulnerability, we ask you to file a vulnerability report by completing this form or sending an email to* ***security@acmeinc.com****. Please include all*

[19] Bug bounty platforms buy researcher silence, violate labor laws, critics say, https://www.csoonline.com/article/3535888/bug-bounty-platforms-buy-researcher-silence-violate-labor-laws-critics-say.html, last visited on 05/24/2021

necessary details for us to be able to understand the impact and reproduce the vulnerability. We also ask you to provide and justify a CVSS score for the vulnerability. If we have trouble understanding or reproducing the vulnerability, we will contact you via email.

To ensure confidentiality, we encourage you to encrypt all communication to us via e-mail. We are equipped to receive messages encrypted using S/MIME. A copy of the certificate that can be used to encrypt your email can be found here.

The email address security@acmeinc.com is intended for reporting security vulnerabilities ONLY. For technical support matters, please use the respective help pages of the product or service in question.

After a vulnerability has been reported, Acme Inc. will acknowledge the receipt within 3 business days. To the best of our ability, we will work to confirm the existence of the vulnerability and initiate remediation. Acme Inc. commits to a transparent process and will update you on the status of the vulnerability as needed, but certainly when it has been remediated.

Scope The targets which are eligible for testing:

This policy applies to the following target systems and services owned by Acme Inc.:

1. **.acmeinc.com*
2. *www.acmeinc-staging.com*
3. *api.acmeinc-staging.com*
4. *Acme Inc. Android application [link]*
5. *Acme Inc. Ios Application [link]*

Any target not listed above should be considered out of scope for the purposes of this bug bounty program, thus any testing of these targets is prohibited.

Excluded Vulnerability Types A list of typically excluded vulnerability types. The list should be modified and adapted based on the needs of the targets and organization's security program.

The following vulnerability types are specifically excluded from the bug bounty program:

- *Descriptive error messages (e.g. Stack Traces, application or server errors)*
- *HTTP 404 codes/pages or other HTTP non-200 codes/pages*
- *Banner disclosure on common/public services*
- *Disclosure of known public files or directories, (e.g. robots.txt)*
- *Clickjacking and issues only exploitable through clickjacking*
- *CSRF on forms that are available to anonymous users (e.g. the contact form)*
- *Logout Cross-Site Request Forgery (logout CSRF)*
- *Presence of application or web browser 'autocomplete' or 'save password' functionality*

- *Lack of Secure and HTTPOnly cookie flags*
- *Lack of Security Speedbump when leaving the site*
- *Weak Captcha / Captcha Bypass*
- *Username enumeration via Login Page error message*
- *Username enumeration via Forgot Password error message*
- *Login or Forgot Password page brute force and account lockout not enforced*
- *OPTIONS / TRACE HTTP method enabled*
- *SSL Attacks such as BEAST, BREACH, Renegotiation attack*
- *SSL Forward secrecy not enabled*
- *SSL Insecure cipher suites*
- *The Anti-MIME-Sniffing header X-Content-Type-Options*
- *Missing HTTP security headers*

Restrictions Defines the various restrictions of the vulnerability disclosure policy:

> *Any services not explicitly listed in the in-scope section of this policy are considered* ***out of scope****, and testing on these targets is prohibited. Any active research and testing on out-of-scope targets is considered a violation of this policy. If it is unclear whether a particular system falls within the scope of this policy, please contact us first at security@acmeinc.com.*
>
> *In addition to the excluded targets and services, the following actions are* ***prohibited*** *while testing as part of this vulnerability disclosure policy:*
>
> 1. *Social Engineering*
> 2. *Physical attacks against Acme Inc. infrastructure*
> 3. *Denial of service attacks*
> 4. *Automated scanning*
> 5. *Interaction with legitimate user data*
>
> *Finally, Acme Inc. excludes the following individuals from participating in this vulnerability disclosure program:*
>
> 1. *Individuals residing in a sanctioned country designed as such by local or international law*
> 2. *Employees and contractors of Acme Inc.*
> 3. *Individuals under the age of 18*

Vulnerabilities on third-party software This section handles the situation when vulnerabilities have been identified in third-party software used by the organization:

> *Acme Inc. uses various services and products from third parties providers. Please note that third-party systems are not covered by this policy and active testing cannot be authorized. Any vulnerabilities identified in third-party systems are not considered in scope as part of this vulnerability disclosure policy. We ask that you report any*

identified vulnerabilities directly to the third party.

*For recently disclosed vulnerabilities on software used by Acme Inc., this policy applies a „patching period“ delay. As such, reports for recently disclosed vulnerabilities will not be accepted until **30 days** since the initial disclosure have passed.*

Hall of Fame This section is used to credit whitehat hackers for finding valid vulnerabilities:

Acme Inc. would like to thank anyone for participating in our bug bounty program. We acknowledge the hard work and responsible behaviour of the security community. Visit the following page for a list of individual contributors. If you have found and reported a valid vulnerability, our team will automatically add your contribution to the hall of fame.

Public disclosure This section handles requests to publicly disclose reported vulnerabilities:

Acme Inc. is committed to the protection of customer data. To that end, any unauthorized disclosure of vulnerability information to the public will be considered a violation of this policy. If you would like to publicly disclose a vulnerability, explicit permission is required by contacting security@acmeinc.com. Public disclosure requests will be evaluated on a case-by-case basis and the full contents of the writeup must be provided ahead of disclosure.

Legal notice This section covers the legal aspects of running the bug bounty program, taken from academic research and public examples[20]. It should be adjusted and adapted to the organization's needs:

To encourage responsible disclosure, we will not pursue civil action or initiate a complaint to law enforcement for security research and vulnerability disclosure activities conducted in consistence with all this policy guidelines. We consider security research and vulnerability disclosure activities conducted in consistence with this policy and guidelines „authorized“ conduct under the Computer Fraud and Abuse Act, the DMCA and applicable anti-hacking laws such as Cal. Penal Code 502(c). We waive any DMCA claim against you for circumventing the technological measures we have used to protect the applications in scope.

Please understand that if your security research involves the networks, systems, information, applications, products, or services of another party (which is not us), that third party may determine whether to pursue legal action. We cannot and do not authorize security research in the name of other entities.

You are expected, as always, to comply with all applicable laws.

[20] Template 2: Explicit safe harbor without good faith violations, https://github.com/EdOverflow/legal-bug-bounty/blob/master/templates/safe_harbor.md, last visited on 05/24/2021

5.2 Using a Bug Bounty Platform

The alternative to the institutional approach is using one of the available bug bounty services platforms. As shown in section 4.3, there are a number of providers in the market with different offerings and platforms. The obvious advantage of using a bug bounty platform is to benefit from platform functionality such as integration tools, crowd management, reward delivery etc. Furthermore, these third party providers already have established communities of whitehat hackers which are ready for testing an organization's targets[72]. If triage services are purchased, these platforms can also filter out most of the noise that comes from invalid reports, effectively reducing the effort on the organization side. However, Bell et al.[72] argue that using a bug bounty platform comes at the cost of losing control. They say that outsourcing the bug bounty program means that others will be representing and making decisions about the organization's security. Additionally, the third party will have knowledge of vulnerabilities and internal security, making them a liability for the organization.

The first step for using a bug bounty platform is to pick the right vendor. As shown in section 4.3, certain vendors have better dispositions for running certain types of testing. For example, if the goal is to test a large scope of targets using a wide pool of testers, Bugcrowd and HackerOne are favorites due to their large crowds. For targets which require trusted whitehat hackers and tight control over the testing process, Synack and Cobalt are providers with a focus in this area. Some vendors may also have limitations that would prevent enrolling certain targets. For example, production targets may have monitoring tools in place which would raise alerts of an attack while crowdsourced testing is being performed. This requires the vendor to have a vpn or proxy service, to separate and whitelist traffic generated from the bug bounty program. Vendors that cannot fulfil this requirement can be disqualified immediately. To properly decide the best vendor, a selection based on a set of criteria should be performed. Some useful criteria when selecting a bug bounty vendor are:

1. Product offerings (private programs, public programs, crowd-sourced penetration testing)
2. Services offerings (triage service, retest service, reward delivery, crowd management)
3. Integrations (api, slack, jira)
4. Platform functionality (notifications, alerts, reporting, analytics)
5. Cost

It is recommended to add additional criteria which is specific to the business impact of the targets. The personnel involved in the criteria selection and final decision should be the stakeholders as mentioned in section 5.1.7. Additional personnel can be included as needed.

Once the vendor has been selected, they should be tested for their performance by running a trial. This can be a time-bound private bug bounty program with a limited number of testers. It is recommended to select a target which is easily accessible and requires little setup to run the bug bounty program. More detailed instructions about picking the right target are given in section 6.2.1. If financially possible, it is recommended to run a trial program with multiple vendors. This allows for direct comparison of performance, and can be included in the vendor selection criteria. When running a trial program, this thesis suggests to consider the following performance metrics:

1. Effort required to enrol targets.
   a) How easy can targets be added and removed?

   b) How are rules of engagement managed for each target?

2. Platform functionality and integrations.
   a) Is the platform intuitive to work with?
   b) Do all advertised integrations work as expected?

3. Speed and professionalism in services delivery.
   a) Are services completed in time?
   b) Is the team professional and capable?

4. Crowd performance and behaviour.
   a) Are whitehat hackers finding interesting vulnerabilities?
   b) Are whitehat hackers staying within scope and following the rules of engagement?

5. Cost
   a) What is the cost for using the platform and its services?
   b) Are there minimum requirements for reward amounts?

If the trial run is successful and the vendor performs at the expected level, they can be fully onboarded. This includes actions such as: setting up a target enrolment process, specifying points of contact, reward guidelines, signing contract etc.

Enrolment process Since enterprise organizations typically will enrol more than one target, a process of enrolment needs to be defined. It should be clarified with the vendor how targets can be added and removed to the platform, and how rules of engagement are managed. It is particularly important for exclusions and restrictions to be set on a per-target basis, since targets may have different requirements. If multiple targets use the same restrictions, one option is to create a base „restrictions and exclusions“ segment, where individual per-target items can be added as needed.

Point of contact Since using a bug bounty platform is a partnership, ongoing communication is a key success factor. Each side should specify the main point of contact and set up an ongoing cadence of meetings. There are a number of actions within the bug bounty platform which require input from the organization. For example, sometimes the business impact is not clear and a member of the organization's security team needs to make that call. Also, unless agreed otherwise, the normal process for deciding the reward amount and issuing the reward within the platform is left to the organization running the program.

Reward guidelines Typically, reward amounts are advertised in the rules of engagement. This is an additional item that needs to be synchronized with the bug bounty platform vendor. It is important that rewards can be advertised on a per-target basis, since not every target has the same impact to the business. In case there are a high number of targets, it is recommended to separate them into categories and advertise rewards for each category separately.

After onboarding has been completed and testing is under way, it is recommended for an organization to monitor the performance metrics and activity of the crowd. As mentioned in section 5.1.7, there are various ways to incentivize the crowd and direct their testing efforts. Typically, bug bounty platforms have dedicated personnel for managing the crowd and will work with the organization to achieve the best performance.

6 Improving Enterprise IT Security With Crowdsourced Offensive Security

In the previous chapter, two options for integrating crowdsourced offensive security in enterprise IT have been presented. It was shown that with an institutional approach, it is necessary to create rules of engagement for the bug bounty program. The individual rules were discussed, and a template is provided which can be used and modified by enterprise organizations. When using a bug bounty platform, the advantages and disadvantages were discussed, along with guidelines for selecting a vendor.

In this chapter, it will be shown how enterprise IT Security can be improved by using crowdsourced offensive security. As shown by Bell et al.[72], running a bug bounty program requires careful consideration of aspects such as staffing, rules of engagement, processing reports, issuing rewards, etc. This chapter will offer guidelines for these aspects from an enterprise perspective, and show the value added by crowdsourced offensive security.

As shown in section 4.2, metrics play an important role in measuring the success of a bug bounty program. Quantitative analysis such as the signal-to-noise ratio provide insight into the value generated by vulnerability reports in comparison to the effort required to manage it. Qualitative metrics on the other hand deal with accuracy of vulnerability reports, and efforts required to understand the root cause. These and other metrics need to be measured and analyzed to improve the effectiveness of the bug bounty program.

Another important detail is selecting and preparing targets for crowdsourced security testing. As shown in section 5.1.1, not every target is ready to be tested using bug bounty, and an enrolment process is required to select which ones are appropriate. Additionally, certain targets such as non-production ones are generally safer to test via the crowd. In either case, a target should be prepared for crowdsourced testing and requires maintenance as testing commences.

Finally, vulnerability reports must be received and processed. This includes actions such as filtering out invalid reports, verifying vulnerabilities by reproducing them, deciding and issuing rewards etc. All of these processes enable an enterprise organization to utilize crowdsourced offensive security for improving their overall system security.

6.1 Defining Success Metrics

This section is going to define success metrics that need to be measured and monitored when running a bug bounty program. As shown by Ransome et al.[88], defining success metrics is a critical component in software security which tracks costs and provide significant help in various return of investment calculations. For bug bounties, some of these metrics have been defined in section 4.2, such as the signal-to-noise ratio. However, other metrics can be of interest. For example, the total number of vulnerability reports and total reward amount can give an idea on return of investment compared to penetration testing. Another example is to look at the most common types of vulnerabilities reported, which may give indication on weaknesses in certain areas of development.

This work suggests the following metrics to be defined and measured as part of running a bug bounty program:

1. Total number of vulnerability reports
2. Signal-to-noise ratio
3. Average communication messages per report
4. Total amount of rewards
5. Average amount of reward
6. Average time to reward
7. Average time to fix
8. Number of unique participants
9. Number of rewarded participants
10. Top vulnerability categories
11. Total enrolled applications

Each of these metrics is defined in the paragraphs below, and a formula for measuring them is given.

1. Total number of vulnerability reports This metrics records the total number of vulnerability reports against the target, regardless of the outcome. It serves as an indicator of the crowd's interest in a target, and the amount of testing being performed against it. This metric can be used to direct crowd efforts against desired targets, by modifying the rules of engagement as shown in section 5.1.1.

$$\text{Total vulnerability reports} = count(\text{vulnerability reports})$$

2. Signal-to-noise ratio As defined in section 4.2.1, the signal and noise ratios represent the ratio of valid and invalid vulnerabilities compared to the total number of reports:

$$\text{signal} = \frac{\text{valid reports}}{\text{total reports}} \times 100 \qquad \text{noise} = \frac{\text{invalid reports}}{\text{total reports}} \times 100$$

These are considered some of the main health metrics of a bug bounty program, showing the actual value gained and overhead. If there is a high percentage of noise, it could mean that whitehat hackers are not motivated correctly to submit quality reports, or that the rules of engagement are confusing. Generally, an organization should attempt to increase the signal and decrease the noise as much as possible, to achieve maximum productivity with little overhead.

3. Average communication messages per report The average communication messages is a metric that shows the amount of back and forth between the whitehat hacker and triaging team. This metric indicates the effort required to verify the vulnerability reports. Generally, a higher number indicates that researchers are not reporting all the necessary details, which could be a result of bad communication in the rules of engagement or inappropriate design of vulnerability submission form.

$$\text{Average communication effort} = \frac{sum(\text{vulnerability reports})}{(\text{total reports})}$$

4. Total amount of rewards The total amount of rewards is simply the sum of all individual rewards issued as part of the bug bounty program. It shows the value gained and can be a good metric to compare against results and cost of other offensive security testing. Additionally, this metric can be associated with the signal metric of the bug bounty program, to understand the return of investment in terms of vulnerabilities identified for the cost incurred.

$$\text{Total amount of rewards} = sum(\text{individual rewards})$$

5. Average amount of rewards As the name implies, this metric is the average amount of reward issued for valid vulnerabilities. This can be an important metric when calculating forecasts in terms of costs and annual budgets. Additionally, the average reward is typically published by bug bounty platforms such as Bugcrowd[70] and HackerOne[9]. In order to get the attention of the whitehat hackers, it is recommended to have an average reward similar to the bug bounty platforms, and advertise the reward to the crowd.

$$\text{Average reward} = \frac{(\text{total rewards})}{(\text{total reports})}$$

6. Average time to reward This metric calculates the average time required to issue a reward for a vulnerability report. It is calculated from the moment the vulnerability report is sent, to the moment it is verified and a reward is attached to it. Apart from the valuable information on speed of vulnerability verification by security engineers, it is highly recommended to publicly advertise this metric as it sets expectations with the crowd around waiting.

$$\text{Average time to reward} = \frac{sum(\text{time to reward valid vulnerabilities})}{(\text{total reports})}$$

7. Average time to fix In this metric, the average time required to remediate a vulnerability is measured. It is calculated from the moment the vulnerability report is verified and passed to the appropriate team, to the moment it has been remediated and deployed. This is a crucial measurement and not only shows the speed of improving the security of an enterprise organization, but keeping the value low will also result in less duplicate submissions and as a result, a more engaged crowd of testers.

$$\text{Average time to fix} = \frac{sum(\text{time to remediate valid vulnerabilities})}{(\text{total reports})}$$

8. Number of unique participants The number of unique participants is a great metric to understand the diversity and size of the crowd taking part in the bug bounty program. Generally, this number is desired to be as high as possible. As shown in section 4.2.2, there are various ways to incentivize participation which leads to increase this number. For retaining participants, previously mentioned metrics such as the average time to reward and average reward amount are important.

$$\text{Unique participants} = count(\text{individual participants})$$

9. Number of rewarded participants This metric will show the number of participants that have reported at least one valid vulnerability and received a reward for it. The importance of this number is for general crowd health, and making sure that whitehat hackers are benefiting from the bug bounty program as is the organization. After all, crowdsourced offensive security relies on a healthy and participating crowd to contribute with their security testing, and without return of investment participants will leave quickly.

$$\text{Rewarded participants} = count(\text{individual participants rewarded})$$

10. Top vulnerability categories By looking at the top types of vulnerabilities being reported, a general feel for weaknesses in development can be identified. This could allow organizations to better understand their strengths and weaknesses in IT Security, and work on mitigation strategies. For example, if many client-side (valid) vulnerabilities are being reported, it could indicate that developers need better training on that front. Or if the bug bounty programs identifies a lot of sensitive information exposed, the attack surface needs to be better managed.

$$\text{Top vulnerability categories} = count(\text{vulnerability reports per category})$$

11. Total number of enrolled targets As mentioned in section 5.1.1, enterprise organizations typically have a rapidly changing IT infrastructure with many potential targets. Ideally, all of these targets are put in scope for the bug bounty program, and scrutinized by crowdsourced security testing. However, section 5.1.1 also clarified why some targets may have challenges with this type of testing and are left out. By measuring the targets which have been enrolled in the bug bounty program of an enterprise organization, an overall picture of security testing coverage by the crowd can be obtained. This metric can also serve to set internal goals for target enrolment by the organization. Since enterprise organizations may have many smaller independent departments, the bug bounty program must be advertised internally for them to join. Some of the metrics mentioned in this section can help to achieve this, and as a result, increase coverage of crowdsourced security testing within the organization.

$$\text{Total number of enrolled targets} = count(\text{number of targets})$$

These metrics should provide a good baseline for understanding the health and progress made by a bug bounty program. Additional metrics can be added or particular ones can be removed if an organization is not interested in that number.

It is also recommended to track trends in these metrics, as they can provide valuable insights. For example, a decrease of a particular vulnerability category can hint at improvements being made in the security training of engineers. Increases in the number of participants show a greater interest in the bug bounty program, possibly as a result of a successful marketing campaign.

6.2 Selecting and Preparing Targets for Testing

As established in section 5.1.1, not every target is suited for crowdsourced security testing. For example, legacy systems may not be able to withstand against modern security testing tools and crash. Also, crowdsourced security testing on targets without any kind of prior security testing could result in a high number of discovered vulnerabilities, and as a result, much higher cost than other security testing methods. Therefore, a selection process is required which will serve two main purposes: selecting appropriate targets for crowdsourced security testing, and determining whether a target is ready for it.

The selection process will find suitable targets for crowdsourced security testing within an enterprise organization. It takes into account a number of factors such as whether prior testing has been conducted, what was the result of prior security testing, can the target be reached from the internet, is it a production or staging environment etc. The selection process can be used when creating a bug bounty program for the first time, or when crowd engagement starts to drop and fresh targets need to be introduced. Section 6.2.1 will provide details on how the selection process works.

Once a target has been selected for crowdsourced security testing, it must be prepared for the upcoming test and maintained during testing. Preparation actions include things such as: deploying a testing instance, creating test credentials, populating test data etc. Some of these actions may need to be repeated while testing is under way. For example, more test credentials may be required because of interest by whitehat hackers, or the target must be reset and cleaned up from time to time. Section 6.2.2 and 6.2.3 will go into details for these processes.

6.2.1 Selecting the Appropriate Targets for Crowdsourced Testing

As established in section 4.2, bug bounties effectively complement other security testing methods such as penetration testing. Hence, this and other factors need to be considered when deciding on targets for crowdsourced security testing. Figure 6.1 shows a recommended process for selecting targets to be added to a bug bounty program. The external nature of whitehat hackers in crowd-

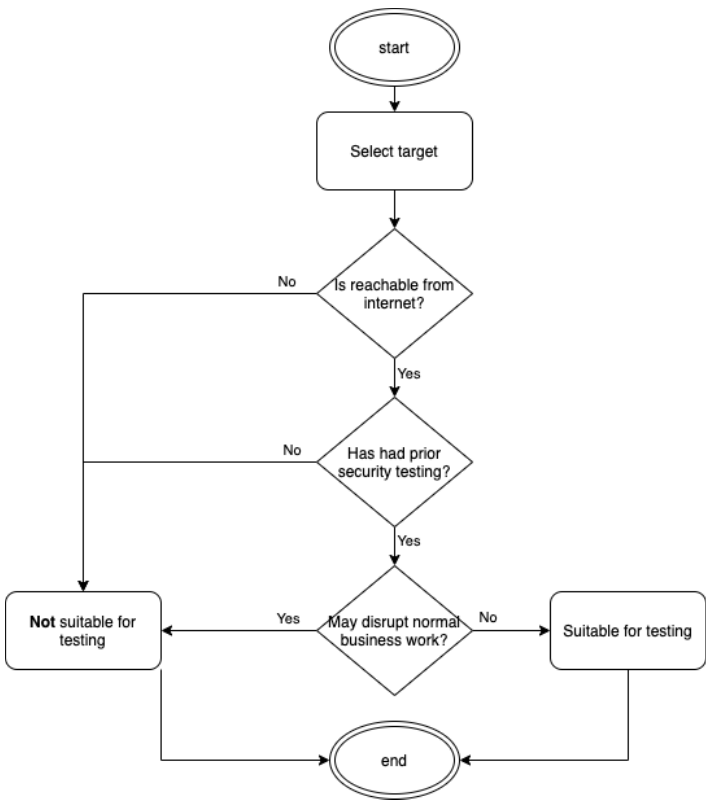

start
Select target
Is reachable from internet?
No
Yes
Has had prior security testing?
No
Yes
May disrupt normal business work?
Yes
No
Not suitable for testing
Suitable for testing
end

Figure 6.1: Target selection process

sourced security testing requires the target to be reachable from the internet. If the target requires access to the internal work, it is not a good candidate for this type of testing. Because the target is internal, it is not exposed to external attacks as other targets are. Hence, traditional security testing such as penetration testing and red team engagement may suffice in this scenario.

Another reason to exclude a target from crowdsourced security testing is if it was not assessed before using other offensive security testing methods. As shown in section 5.1.1, such a target may contain a high number of vulnerabilities and result in exponentially higher cost compared to penetration testing. Prior security testing may also identify issues that need to be considered when security testing is performed, such as managing credentials for login.

If the target is a production environment, crowdsourced security testing may result in disruption of normal business continuity. Because they are not bound to a testing methodology, whitehat hackers will test any functionality and fill out any form. This may result in an increase of false positive requests to the business and disrupt or delay legitimate ones. There are ways to mitigate these issues which are shown in the following section.

Finally, some targets may have high confidentiality requirements such military systems or critical infrastructure. Knowledge of the existence of these systems or public access may provide real threat actors an opportunity to investigate and analyse the systems. For these targets, alternative offensive security testing methods should be considered.

6.2.2 Preparing the Target for Testing

Preparing a target for crowdsourced security testing is of similar importance as selecting the appropriate target. After a target has been selected, necessary preparations should be made before going live. This should include at least the following actions:

- Deploy a test instance (if possible)
- Prepare testing credentials
- Prepare test data

Deploy a test instance (if possible) As shown by Schanes et al.[89], security testing in large IT infrastructures is commonly done on test environments. While this may have limitations and not fully emulate operational environments, there are also benefits. First, when using a test environment there is no impact if the target crashes and goes offline. As such, denial of service testing can be included in scope. Additionally, credentials can be more easily managed in test environments, and the instance can be reset if something goes wrong. A test environment has also the benefit of being decoupled from (sensitive) production data in case whitehat hackers find critical vulnerabilities. Of course there is additional management overhead when setting up a separate instance just for security testing, and there are cases when this is not possible at all.

Prepare testing credentials To achieve the most coverage, testing credentials should be provided for whitehat hackers. If the target allows self-signup, then there is no need to create credentials. However, there are cases where where registering is not possible due to it being disabled, requesting a credit card or social security number etc. In these situations it is beneficial to directly provide credentials to the crowd. Distribution of them can be done by the support representative, and the process for requesting credentials can be stated in the disclosure policy. For targets which require multiple roles, a set of credentials for each role should be provided. Again, the goal is to allow whitehat hackers to test all available functionality for security vulnerabilities. This increases coverage from the organizaiton perspective, and also provides more attack surface for whitehat hackers to find vulnerabilities and get paid for their efforts.

Prepare test data As has been established, interaction with real customer data is generally undesired. Therefore, if possible the target should be populated with test data for security testing. Test data provides a clean separation and enables security testing without disturbing legimiate users.

For example, if the target is a hotel booking system, then a test hotel should be established for security testing. When the target is a test environment, testing data may be required for whitehat hackers to be able to interact with and test all functionality.

Additional preparations such as the ability to reset the test environment may be useful. The organization can also decide to involve the marketing team and advertise the scope increase, which picks the crowd's attention back to the program.

6.2.3 Maintaining the Target Being Tested

As testing by whitehat hackers is under way, various issues can arise that must be repaired. Therefore, a target tested by bug bounty should be maintained regularly. One such issue may be invalid data stored by the crowd. Because the number of testers is large and many of them use automation tools, the test instance can grow large with dummy data. Hence, regular cleanup is recommended.

As has been established, the target may also crash during testing. Therefore, having monitoring systems that raise alerts should be considered as part of target maintenance. For production systems, such monitoring may already exist and the bug bounty operator only requires to subscribe to those notifications. If no such mechanism exists, it is recommended to create one.

Target maintenance is also considered resetting existing credentials and creating new ones for distribution. In the previous section it was shown why credentials are important. Since a large set of whitehat hackers will be engaged in testing the target, many of them will require credentials. Others will stop testing the target after some time, and those credentials can be recycled for new testers.

6.3 Launching the Bug Bounty Program

With rules of engagement in place and targets selected, the bug bounty program can be officially launched. This can be as simple as publishing the rules of engagement and enabling receipt of vulnerability reports. However, it may make sense to include the marketing team and advertise the launch of the program to attract whitehat hackers. This is especially true for institutional bug bounty programs. When using a bug bounty services provider, typically a process for launching a program and inviting security testers is already established.

As shown by Maillart et al.[6], launching a bug bounty program will result in an initial spike of vulnerability reports. The same is reported in practice by large organizations such as Yelp[1], who experienced a large volume of new reports in the first few days after the public launch. To provide a good first impression and maintain researcher satisfaction, this work suggests to have at least two dedicated security engineers for the first two weeks. If a bug bounty coordinator is appointed, they should also be fully allocated to the success of the program in this initial period. This helps to ensure operational processes are successfully executed and potential issues resolved.

After the initial spike of vulnerability reports, the bug bounty program will begin to stabilise in terms of work overhead. The next phase can be seen as a learning period, where the bug bounty management team shall closely observe how operational processes are functioning, if crowdsourced security testers are reporting issues etc. The learning period will also allow se-

[1] First 100 Days of Yelp's Public Bug Bounty Program, https://engineeringblog.yelp.com/2016/12/100-days-public-bug-bounty-program.html, last visited on 05/24/2021

curity engineers to get used to the process. Observations throughout this period may identify process improvements for efficiency, such as certain tools to use during vulnerability triage. There may also be opportunities to improve the rules of engagement, by excluding certain vulnerability types which are being reported as false positives. It is recommended to make observations and improvements throughout the lifecycle of the bug bounty program, however, the initial period will be more intensive. As the program matures and the organization gains experience, a normal operating mode will become standard.

As defined in section 6.1, metrics should be measured to ensure the success of the bug bounty program. This work recommends that measuring these metrics begins after the initial spike of vulnerability reports. Because the initial volume is not representative of the normal workload throughout the bug bounty program, it is generally not of interest with regards to success metrics. Of course, an organization can decide to measure everything and observe the success of the preparation for the launch. It is important to be aware that if metrics are collected from the start, a longer period of time will be required to gain a true picture of the data. Finally, the process for collecting these metrics must be defined. Generally, the bug tracking system used to maintain valid vulnerabilities can also be used to extract these metrics. However, if it does not have all necessary capabilities, alternatives should be taken into consideration. Collecting the metrics manually will likely result in high overhead and may be subject to errors or process failures.

Finally, an organization should be prepared to temporarily pause the bug bounty program. This can happen for a number of reasons. For example, testing by the crowd may disrupt normal business operations or prevent them completely. An organization may also identify systematic vulnerabilities in their systems, and may decide to pause and evaluate the next steps. The decision to pause the program will likely result in dissatisfaction with the crowd, since it will prevent them from continuing testing after having invested time to get familiar with the targets. Therefore, it is important to have good communication. This work recommends that the following actions are taken if a bug bounty program is temporarily suspended:

1. Update the rules of engagement and state the program is pausing.
2. Disable receipt of vulnerability reports if possible.
3. Message each existing participant.

Updating the rules of engagement will ensure that new participants are aware of the situation. However, existing participants will typically read the rules only once, and then focus on testing. Therefore, it is recommended to also notify all existing participants. It is also important to state the reason for the temporary suspension, and provide a date for the program restart. If such a date cannot be provided, a date for the next communication should be given. This helps to set correct expectations with the whitehat hackers. Finally, all existing vulnerability reports should be triaged and rewarded. Because those vulnerability reports were submitted when the bug bounty program was live, they should be handled accordingly. This information should also be accompanied in the messaging that goes to the crowd, to ensure they will be compensated for the work they have done.

6.4 Processing Vulnerability Reports

Receiving and processing vulnerability reports is one of the main activities when running a bug bounty program, as established in section 4.1.3. This includes activities such as verifying validity of vulnerability reports, issuing rewards, as well as maintaining communication with the reporter.

As shown in section 5.1.1, there are a number of ways to receive vulnerability reports. One common approach is to use an email address and request whitehat hackers to submit a report via email. Another approach is use to use a submission form which has the advantage of defining a structure. By requesting whitehat hackers to fill all fields of the form, information required to process the report can be obtained in a desired format.

By now it has been established that bug bounty programs receive many reports which are invalid[15]. Hence, it is needed to verify the validity of them. Typically, this includes reproducing the researcher claims and assessing whether it actually is a vulnerability (that is, where it has security impact). This process filters invalid reports and effectively the noise of the program.

An important aspect in handling vulnerability reports is communication with the vulnerability finder. In order to avoid confusions and researchers asking for an update of their report, a steady and transparent communication stream should be maintained. As shown by Malladi and Subramanian[14], transparency and responsiveness are amongst the key factors for building trust with the whitehat hacker community.

6.4.1 Triaging Vulnerability Reports

Once a vulnerability has been reported, it must be verified for validity. As can be seen above, this typically includes reproducing the vulnerability and determining whether it has security impact. This is typically the responsibility of security engineers involved with the bug bounty program.

In section 4.1.3, the typical flow of a vulnerability report as part of a bug bounty program is shown. It includes the following activities:

1. Check if the vulnerability report is in scope
2. Check if the same vulnerability has already been reported
3. Check if the vulnerability report is a valid security issue
4. Reproduce the claims in the vulnerability report

The first three activities are also known as filtering out the noise or invalid reports. As established in section 4.1.3, these are reports which are completely irrelevant, false positives or out of scope for the bug bounty program. There are various reasons why crowdsourced security testing is accompanied with invalid vulnerability reports: incorrect research, lack of validation by the finder etc. Filtering out invalid reports can be time consuming, which is why bug bounty platforms typically apply penalties for them[2]. When enterprise organizations use a bug bounty platform, they can purchase triage services[3] which encompass submission processing, and as part of it clear out the noise. In an institutional approach, organizations must handle this process internally.

Once a vulnerability report passes the initial noise filter, its claims need to be verified by the bug bounty operator. This includes following the steps provided by the whitehat hacker, and checking if the results match their claims. The verification process can further identify invalid reports, if the claims are false or the results have no security impact. If the claims are verified and indeed a security issue is identified, the vulnerability can be filed with the internal bug tracking software of

[2] Report States, https://docs.HackerOne.com/programs/report-states.html, last visited on 05/24/2021
[3] HackerOne Services, https://www.hackerone.com/services, last visited on 05/24/2021

the organization.

In the following sections, each of these activities will be described in more detail. Typically, all these activities are performed by security engineers. Triaging vulnerability reports as part of a bug bounty program requires IT Security skillsets, even for filtering out noise. That is because an inexperienced person may discard valid vulnerabilities as noise, resulting in the loss of value and risking exposure of the target organization.

1. Check if the vulnerability report in scope. This check confirms whether the vulnerability report complies with rules of engagement. This requires performing two sub-checks: check if the target is in scope, and check if the vulnerability type is allowed. Checking whether the target is in scope is done by comparing the target of the vulnerability report with the in-scope targets. This is a fast process and therefore suggested to be done in the beginning. If the check fails, the vulnerability can be discarded and the reporter notified. However, there are also situations when this check is more time consuming. For example, if the scope of the bug bounty program allows reports on any assets owned by the organization, the security engineer performing this check must first ensure whether the target is owned by the organization or not. Another situation may be an out-of-scope report by a whitehat hacker who is simply acting responsibly, which should be acknowledged. If the target check is passed, the vulnerability type must be identified and compared against the excluded vulnerability types as specified in the rules of engagement. Determining the vulnerability type is easy, since whitehat hackers typically use it to claim security impact. The comparison against the list of excluded vulnerabilities is also a fast process, and allows to easily discard vulnerabilities which are not desired.

2. Check if the vulnerability has already been reported. While duplicate vulnerability reports are not necessarily noise, they do not bring additional value to the bug bounty operator. Hence, they are considered noise. Generally, comparing a vulnerability report against previous report is a fast process. However, if a large number of vulnerability reports have been received in the past, the time required to check them can increase drastically. Another factor is the number of reported vulnerabilities which have not been patched yet. If the number is high, there will be more duplicate reports which are noise for the organization and generally result in dissatisfaction with the crowd. Finally, sometimes the same vulnerability may get rewarded multiple times. For example, if a vulnerability has been remediated as a result of a previous report, but has appeared due to a regression, it should be rewarded as a new finding. Another example is if a whitehat hacker has found a bypass for a previously fixed vulnerability, it should not be treated as a duplicate. A good rule to follow is asking the question whether the vulnerability report provides value to the organization. In this situation, without the vulnerability report for the reappearance or bypass of the (previously fixed) vulnerability, the organization would not have known about it. Hence, this should be considered a valid vulnerability and rewarded accordingly.

Check if report is invalid. As has been established, some vulnerability reports are simply spam and provide no useful information. In order to determine this, a security engineer needs to read and understand the whole report. Therefore, the time required to perform this action may vary depending on the length of the vulnerability report. An invalid vulnerability report can be discarded, however, the reporter should be notified about it. Generally, crowdsourced security testers will not submit spam vulnerability reports for malicious reasons. Instead, it is much more likely that they do not understand the application, or lack the relevant security experience. If possible, a quality score should be maintained for each report. This allows to estimate the likelihood of a report's validity based on the history of the reporter. Bug bounty platforms generally have this built in their platforms, and profit from a large history of reports across their programs. They also publish lists of top whitehat hackers, which can be used by an organization to determine report accuracy.

Check if report is reproducible. Finally, if all previous checks have passed, a vulnerability must be reproduced. For reproduction, security engineers will use information provided by whitehat hackers to recreate the vulnerability internally. This is a time consuming process and may have a number of outcomes. For example, there are situations when incomplete information is provided by the reporter. This leads to an unsuccessful attempt at reproducing the claim by the security engineers, and they must communicate with the whitehat hacker to ask for additional information. This situation will also add overhead to the next time a security engineer will attempt to reproduce the report. That is because the context around the vulnerability needs to be rebuilt, and the claims reverified . The importance of collecting as much information as possible during the initial reporting is emphasized in section 5.1.1. An additional situation may arise when reproduction results in discarding the vulnerability report. This may happen if whitehat hackers make wrong assumptions about particular functionality. For example, a security tester may change security settings on their account, and later report a vulnerability claiming they found a weakness. These situations may be time consuming if the tester fails to report all details, and security engineers attempt to reproduce a false claim. Sometimes it may require research on the security engineer's behalf to understand what is causing the false alarm with the whitehat hackers. In these situations, security engineers must communicate and explain why the vulnerability report has limited or no impact, and as a result, it does not qualify for a bug bounty. At last, if the vulnerability is reproduced and the claims are found to be true, it can be filed with the internal bug tracking software of the organization. Additionally, the process for issuing a reward can be initiated.

In public bug bounty programs there may be a lot of incoming vulnerability reports. Therefore, organizational activities such as prioritizing and resource allocation are needed. In institutional bug bounty programs this may be a difficult task without support by proper software. On the other hand, bug bounty platforms usually have this built in and provide functionality such as sorting, searching, altering on old reports etc.

As can be seen, processing reports can be a time consuming process that scales with the number of incoming submissions. When enterprise organizations use a bug bounty platform, they can purchase triage services which help to deal with such a scaling workload. In an institutional approach, organizations must handle this process internally. In section 4.2, a number of ways to incentivize high quality reports have been shown. Zhao et al.[15] suggest that clearer instructions in the rules of engagement can improve the quality of submitted vulnerability reports. Malladi and Subramanian[14] suggest that a higher reward for high-quality reports will drive results due to the financial motivation of security testers. Another form of reminding the crowdsourced security testers is to implement a „speedbumb“ before they are able to file a report. This can be used to remind the security tester that the rules of engagement must be followed. Other measures could be providing an example report, placeholder text in the report form, or educational material such as blog posts or videos.

Another possibility is to keep the bug bounty program as invite-only, and allow participation only to selected individuals. This is a common approach for lowering noise in bug bounty platforms[4]. In an institutional approach, the lack of historical information about whitehat hacker quality history makes running a private bug bounty program harder. Instead of this approach, organizations may have two options at disposal: run a public program without rewards and a private program

[4] Reducing Noise in Crowdsourced Security, https://www.bugcrowd.com/blog/reducing-noise-in-crowdsourced-security, last visited on 05/24/2021

with rewards simultaneously, and only invite researchers with a history of good quality reports to the private program. The second option is to look at the history of whitehat hacker's report quality, and prioritize reports from higher quality researchers. This however creates a problem of duplicate checking, where a new researcher's report may get delayed and known good researchers may get rewarded for duplicate reports.

6.4.2 Issuing Rewards

For triaged vulnerability reports which result in valid security issues and are the first of its kind, rewards must be issued according to the rules of engagement. For this process, the vulnerability severity must be evaluated first. Afterwards, an appropriate reward amount can be determined and transferred to the reporter.

As shown in section 4.2.3, two popular methods for determining the severity is the CVSS system and Bugcrowd's vulnerability rating taxonomy[5]. CVSS[6] is a universal scoring system that produces a numeric value representative of the severity of a vulnerability. It contains three types of scores: base score, temporal score and environmental score. As shown by Munaiah and Meneely[5], temporal and environmental scores are rarely used in practice. Hence, for the purpose of rating a vulnerability as part of a bug bounty program, the base score will likely suffice. The base score is made up of a number of metrics, which are then used in the base score equation to obtain the severity of a vulnerability. The metrics are grouped by different aspects of the vulnerability, and each metric has a list of possible values as shown in figure 6.2.

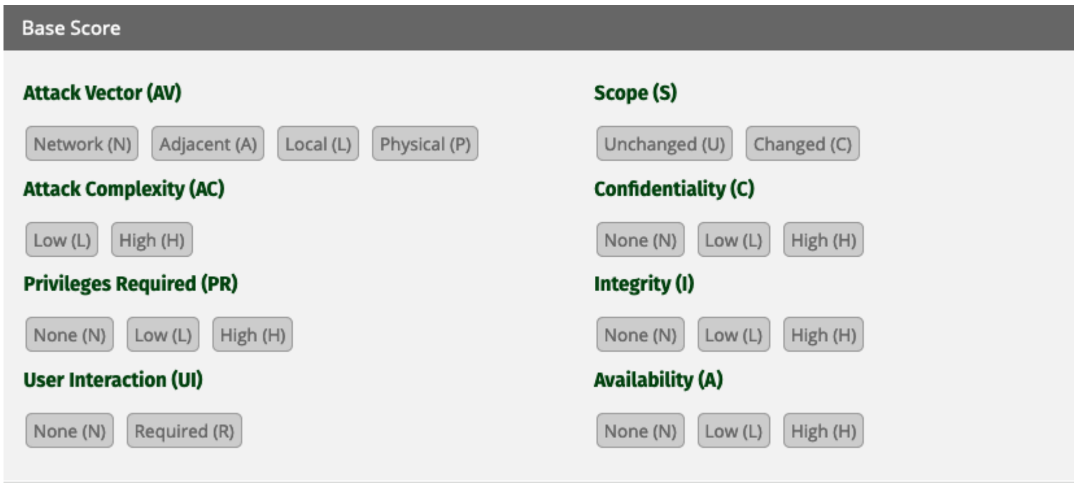


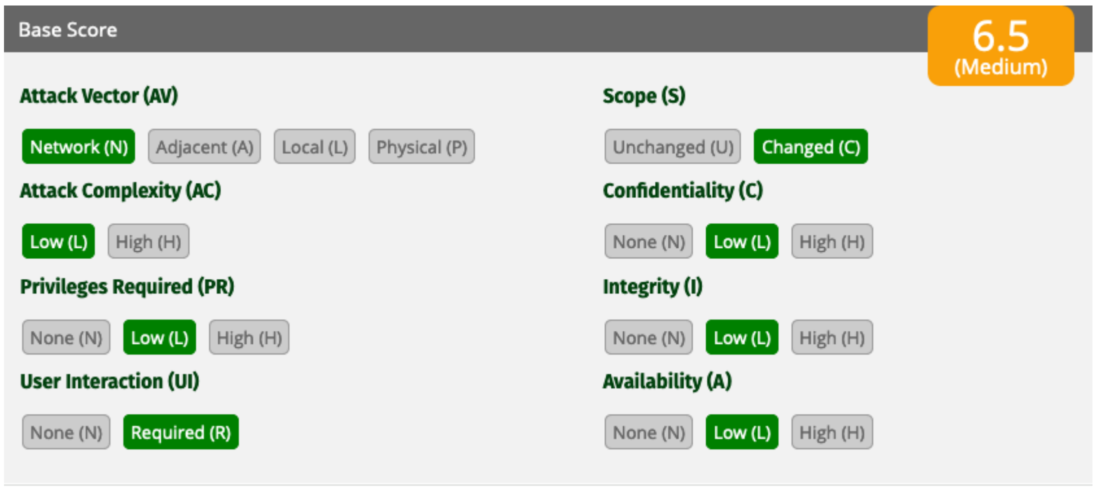
Figure 6.2: CVSS Base Score Metrics

For calculating the score, a CVSS calculator[7] can be used. An example calculation for a reflected XSS vulnerability is shown in figure 6.3. CVSS scores can be converted to a simple system which has only "Low", "Medium", and "High" ranking. This can be useful for determining the reward amount when using reward ranges as shown below. NVD[8] provides a table which can be used to convert base scores into the respective ranking.

[5] Bugcrowd's Vulnerability Rating Taxonomy, https://bugcrowd.com/vulnerability-rating-taxonomy, last visited on 05/24/2021
[6] Common Vulnerability Scoring System SIG, https://www.first.org/cvss, last visited on 05/24/2021
[7] Common Vulnerability Scoring System Version 3.0 Calculator, https://www.first.org/cvss/calculator/3.0, last visited on 05/24/2021
[8] Vulnerability Metrics, https://nvd.nist.gov/vuln-metrics/cvss, last visited on 05/24/2021

The alternative is to use Bugcrowd's vulnerability rating taxonomy[13]. Bugcrowd uses a simple rating system where vulnerabilities can be one of the following severities:

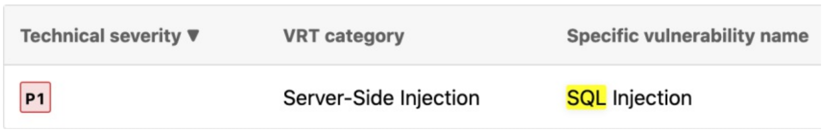
- Critical (P1)
- High (P2)
- Medium (P3)
- Low (P4)
- Informational (P5)


Base Score 6.5 (Medium)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): None (N) [✓ Low (L)] High (H)
User Interaction (UI): None (N) [✓ Required (R)]
Scope (S): Unchanged (U) [✓ Changed (C)]
Confidentiality (C): None (N) [✓ Low (L)] High (H)
Integrity (I): None (N) [✓ Low (L)] High (H)
Availability (A): None (N) [✓ Low (L)] High (H)

Figure 6.3: CVSS Base Score XSS Example

Bugcrowd's vulnerability rating taxonomy is an attempt to streamline and set expectations for both whitehat hackers and bug bounty operators around severity of reported vulnerabilities. They acknowledge[9] that it serves as a base priority, and can be adjusted depending on other factors which may affect severity. It classifies vulnerabilities in categories, subcategories and variants. For each entry, Bugcrowd provides a pre-set severity value. There are also a number of vulnerability types classified as „Varies", which means that the technical severity should be evaluated on a case by case basis. The technical severity for an SQL injection according to Bugcrowd's vulnerability rating taxonomy is shown in figure 6.4



Technical severity ▼	VRT category	Specific vulnerability name
P1	Server-Side Injection	SQL Injection

Figure 6.4: Bugcrowd Vulnerability Rating Taxonomy SQL Injection Severity

Once the vulnerability has been rated with a severity, a reward amount can be decided. There are several ways to select an appropriate reward for a valid vulnerability. For example, in their rules of engagement[10] Facebook commits to a minimum of $500 reward without information for higher

[9] Usage guide, https://bugcrowd.com/vulnerability-rating-taxonomy, last visited on 05/24/2021
[10] Facebook Whitehat, https://www.facebook.com/whitehat, last visited on 05/24/2021

severities. On the other hand, Google provides top level rewards for some vulnerability categories and reward ranges for others[11].

A good starting point for determining reward ranges is the Bugcrowd vulnerability pricing model[12]. They suggest to set reward ranges based on previous testing performed on a target. Table 6.1 shows these suggestions, which are based on Bugcrowd's own vulnerability rating taxonomy.

	Security Maturity Model			
Target	P4	P3	P2	P2
Untested Target	$125	400−600	1,000−1,200	1,750−2,000
Moderately Tested Target	$150	600−750	1,500−1,800	3,100−3,500
Well Tested Target	$200	750−1,000	1,800−2,100	4,500−5,000
Well Hardened Target	$300	1,000−1,250	2,100−2,500	7,500−10,000
Well Hardened Thick Client Binary / Embedded Devices	$500	1,500−2,000	4,500−5,000	$10,000-20,000+

Table 6.1: Bugcrowd Suggested Reward Ranges

Another factor that may impact the decision on reward ranges is the available budget for the bug bounty program. If the budget is low, it is suggested to start with lower reward ranges and see the crowd's interest. Reward ranges can always be increased if little testing is being done by the crowd, but lowering reward ranges will likely result in dissatisfaction. It is also recommended to advertise reward ranges in the rules of engagement, to set researcher's expectations on amounts they may receive for valid vulnerability reports.

Finally, the reward can be delivered to the reporter. Popular methods for issuing the reward are via bank transfer and paypal. Bugcrowd[13] and HackerOne[14] both offer these two options. HackerOne additionally offers payment via Bitcoin. It is also important to communicate the severity and reward amount to the vulnerability reporter. This will provide insight into the decision process and avoid confusion by the reporter. Sometimes a whitehat hacker may argue for a higher impact. These should be evaluated and analyzed whether the reward amount should be adjusted. However, determining the reward is usually at the bug bounty operator's discretion, as long as it complies with the advertised ranges in the rules of engagement.

6.4.3 Communication With the Vulnerability Finder

The importance of communication was established in section 5.1.7, which also showed the consequences of miscommunication. It is therefore recommended to maintain a healthy and continuous communication with the whitehat hacker reporting a vulnerability. If possible, an update should be given for all actions performed in the flow for processing reports as shown in section 4.1.3.

When a vulnerability report is received, a message should be delivered to the reporter acknowledging the receipt. This can be an automatic message with default content. It is also a good idea to

[11] Google Vulnerability Reward Program (VRP) Rules, https://www.google.com/about/appsecurity/reward-program, last visited on 05/24/2021
[12] Bugcrowd's Defensive Vulnerability Pricing Model, https://www.bugcrowd.com/resources/guides/bugcrowds-defensive-vulnerability-pricing-model, last visited on 05/24/2021
[13] Setting Up Payment Methods, https://docs.bugcrowd.com/researchers/payments/setting-up-payment-methods, last visited on 05/24/2021
[14] Payout Preferences, https://docs.hackerone.com/hackers/payout-methods.html, last visited on 05/24/2021

add information about typical response times in this message. Although this is already advertised in the rules of engagement as shown in section 5.1.4, it helps to remind and reemphasise the same in this automated response to the finder.

Once a submission is being processed, communication can continue based on the flow shown in section 4.1.3. For vulnerability reports which are out of scope, a message should notify the researcher about the outcome. It is also recommended to remind the reporter about the scope of the bug bounty program. Similarly, for duplicate reports a message of the outcome should be sent. If possible, some information about the original report should be provided to remove doubt and increase transparency.

If a submission is not reproducible by the security engineer, a message should be sent to the reporter asking for clarification. It is important to provide as much details as possible, and pinpoint exactly where the reproduction fails. In a similar fashion to how vulnerabilities are reported by the crowd, the security engineer working to verify the report should provide their steps of action taken and the results obtained. Screenshots and videos can help to clarify the situation. Additionally, it is recommended to ask the whitehat hacker to retry the vulnerability on a newly created account, to avoid some false positives related to misconfigurations. For submissions which are not applicable or have no security impact, it is recommended to send a message explaining the security engineer's take on the vulnerability. It is important to clarify the situation, but never take things as final. Instead, a chance should be given to the reporter to explain their claims. As has been established, crowdsourced security testers have a diverse skillset, and it could very well be possible that the security engineer lacks expertise to understand the issue. If the whitehat hacker can prove their claims, the vulnerability report should be revised by senior security experts if possible.

Just as security engineers may lack experience in some vulnerability types, the same applies for crowdsourced testers. There can be situations where a whitehat hacker reports a vulnerability and firmly claims it is valid, even if in reality it has no security impact. In this instance, communication can go back and forth many times, taking time away from security engineers. When appropriate, security engineers should end this communication cycle by sending one final message. This message should explain once more why the vulnerability report is not considered valid, and also explain that no further communication related to this matter will be performed.

Finally, as shown by Bell et al.[72], fostering relationships through respectful communication is one of the key success factors of a bug bounty program. This includes prompt responses and keeping the whitehat hacker informed throughout the process of vulnerability verification. It should also be noted that for many of the reporters, english may not be their first language. Hence, respectful and clear communication should be a top priority for any bug bounty operator.

6.5 Risk Mitigation Through Crowdsourced Offensive Security

After having defined the processes for launching and successfully running a bug bounty program, this section will show that enterprise IT Security can improve by leveraging crowdsourced security testing. Specifically, insight into speed of vulnerability discovery, remediation, and retesting fixed vulnerabilities will be provided.

With characteristics such as a large number of testers and different backgrounds, crowdsourced offensive security enables a faster vulnerability discovery cycle. Accessibility to the vulnerability finder also enables quicker remediation, while providing learning opportunities to the development team. Finally, the participating whitehat hackers can be used to test remediations applied for valid vulnerability findings. This can be used to identify potentials errors in remediation, and work to correct them.

The purpose of this section is to show that crowdsourced offensive security is an effective measure that can be deployed by enterprise organizations. As a result, the research question 2 of this thesis as defined in section 1.4 will be answered.

6.5.1 Low Latency Vulnerability Detection

As established in section 4.1, one advantage of using crowdsourced security testing is the number of testers. If participation to the bug bounty program is open to public, there can be thousands of testers from around the world looking to find vulnerabilities in target systems[79]. Due to different locations, timezones and working hours of security testers, for the bug bounty operator this can be expressed as continuous security testing around the clock. It is highly likely that at any point during the day, some whitehat hacker somewhere in the world will be looking and testing the targets for security vulnerabilities.

One benefit of continuous security testing is faster vulnerability detection. Compared to traditional security testing which can be performed once or multiple times a year, bug bounties have a much faster vulnerability detection time[41]. For example, HackerOne[9] reports that 77% of bug bounty programs receive their first vulnerability report within 24 hours. The continuous aspect of bug bounty programs also provides a lower latency for detecting vulnerabilities introduced with new functionality or systems. As a result, the duration of exposure due to a vulnerability is much lower.

A large amount of testers also brings a diverse set of skills with them, as shown by Maillart et al[6]. This increases the likelyhood that some security testers are familiar with the technologies employed by the organization. These testers can use their familiary and experience to discover more vulnerabilities. As shown by Magazinius et al.[7], a large and diverse group of testers with diverse skillsets are likely to discover more vulnerabilities.

As can be seen above, running a bug bounty program provides great benefits to enterprise organizations. Vulnerabilities are detected faster due to continuous security testing by a crowd located around the world. Additionally, the diverse skillset will raise chances of security testing performed by someone who is an expert in the technologies used by the targets. As a result, it can be concluded that bug bounties can be deployed in enterprise organzations, and effectively improve IT Security.

6.5.2 Faster Vulnerability Remediation

Similar to how bug bounties speed up vulnerability detection, they can enable faster remediation. While it may not be obvious at first, this thesis provides the following arguments why vulnerabilities can be fixed faster:

- Communication with vulnerability reporter.
- Continuous remediation leads to more experience.
- Insight into discovery process.

Communication with vulnerability reporter. As shown in section 6.4.3, maintaining a healthy communication with the vulnerability reporter is a key aspect of crowdsourced security testing.

Engaging the vulnerability reporter during remediation can aid software engineers to better understand the root cause, and as a result, apply remediations faster.

Continuous remediation leads to more experience. Another aspect of bug bounties that may improve remediation speed, is its continuous nature. In traditional offensive security methods such as penetration testing and red-teaming, the development team deals with security vulnerabilities only after every testing cycle. However, bug bounties continuously raise security issues that must be addressed. This ensures that software engineers are dealing with security all the time, thereby gaining valuable experience. The constant nature of security testing by bug bounties keeps software engineers alert and conscious about security implications that may arise as a result of their code. As shown by Poller et al.[90], security audits increase the security awareness of software developers and as such, contribute to better code security.

Insight into discovery process. Finally, the discovery process of whitehat hackers reveals valuable insights for organizations running a bug bounty program. It provides a realistic picture of external attackers, how they approach a target and what patterns they use to test them. Such information can be incorporated into the cybersecurity program of the organization, to strengthen defences and detection capabilities.

6.5.3 Retesting Fixed Vulnerabilities

The last aspect of bug bounties that contributes to better risk mitigation, is the ability to perform retesting of valid vulnerabilities after a patch has been applied. Because the vulnerability finder is easily accessible and communication is available, they can be asked to retest their finding.

Retesting fixed vulnerabilities has two main benefits: it verifies that the original security concern has been remediated, and it offers the reporter an opportunity to find a bypass. It may be possible that software engineers did not understand the root cause of the vulnerability and therefore apply a wrong or incomplete fix. By asking the original vulnerability reporter to retest the issue, both of these situations can be identified and acted upon. Compare this to traditional offensive security testing, where typically remediations are not tested until the next testing cycle and may be missed entirely.

This work recommends to offer some form of compensation for this action, since it is takes effort on the side of the whitehat hacker. This compensation is likely to be a smaller cost to the organization, compared to allocating their internal security team for this task. The reason is that the original vulnerability reporter has context around the security vulnerability, and potentially also a ready environment. This allows for much faster retesting, compared to an internal security engineer that must build the context and prepare the testing environment before they can retest the fix.

7 Enterprise IT Crowdsourced Security in Practice: A Case Study

In this chapter, a case study of creating a bug bounty program for an enterprise organization is presented. This case study is used to practically apply the concepts of this thesis and analyze the results. Based on the results, lessons learned for improving the process will be noted and conclusions drawn.

After introducing the organization, the process of creating a bug bounty program for one of its products is shown. The approach for integrating a bug bounty is given, along with the process and reason for selecting that specific approach. Afterwards, the learnings of creating the rules of engagement are covered. Further sections show the process for launching and operating the bug bounty program, as well as an analysis of the case study. Finally, expert interviews are conducted to evaluate the performance of this thesis.

7.1 Introduction

This case study was conducted in cooperation with an international IT services provider with over 20 years of experience in IT planning, IT infrastructure, IT architecture and software development. The organization is based in Austria, and has developed projects ranging from small to highly complex IT systems with budget in millions of euros.

This organization decided to run a bug bounty program on one of its major products, a solution for electronic health record management. This product is a service package for healthcare providers and insured patients, and provides optimal integration between involved parties. By storing information such as diagnosis, results, therapeutic measures, treatment reports, information on vaccinations etc. it provides an excellent information and decision-making basis for doctors, therapists and pharmacies.

The product offers mobile applications in android and iOS for insured patients, backend services and a software development kit for integration by healthcare providers. A key aspect of this product is the security and privacy of its data and systems. Patients are given the ability to decide who can access their information and to what extent. This is achieved through a high security IT infrastructure, which provides the necessary means to securely store and exchange the data between parties.

7.2 Creating a Bug Bounty Program for an Enterprise Organization

In order to test the security of this product, the organization decided to run a bug bounty program. Due to the high security requirements, the product has been designed with security mind. Additionally, the organization used its internal security team to perform penetration testing on the applications. The decision to run a bug bounty program was made as a complementary security measure, and serves two main purposes:

1. Confirm a high security level of the application and that no low impact issues can be found.
2. Improve security by offering rewards to crowdsourced security testers.

This engagement also serves as a trial for crowdsourced offensive security, and lays the grounds for expanding it to other products.

7.2.1 Selecting the Integration Approach

For the purpose of running a bug bounty program on this specific product, the organization has chosen an institutional approach. Although this thesis recommends taking bug bounty platforms into consideration when deciding the integration methodology, a number of reasons influenced the decision to go forward with an institutional bug bounty program.

As shown in section 5.1.1, one reason to select an institutional approach is to keep complete control of the bug bounty program. Due to the highly sensitive nature of health information stored in this product, it was built as a high security system. Therefore, being able to control all aspects of the bug bounty program was a key factor in the decision. Additionally, having complete control over the program provides flexibility in operational processes and changes. Any latency that may have been generated by bug bounty platforms is not an issue here, since in-house security engineers are familiar with the product and can quickly verify vulnerability reports.

Another reason to chose an institutional bug bounty program is to prevent third-parties from having access to data. As has been established, the product stores critical health data such as diagnosis, results and therapeutic measures. Running a bug bounty program through one of the bug bounty services providers would give them access to information and potential vulnerabilities, which is undesired in this situation.

The final reason for selecting an institutional approach is cost. The expected workload in terms of reported vulnerabilities seems to be manageable due to the high security nature of the system and relatively low attack surface. Additionally, the organization has the necessary resources to manage the bug bounty program with its large internal security team. Therefore, managing the bug bounty program in-house is seen as a more cost-effective option compared to purchasing services from a bug bounty services provider.

7.2.2 Creating the Rules of Engagement

After the integration approach has been decided, the process for creating the rules of engagement has been initiated. The template provided in this thesis has been used as a starting point, with adaptions made as follows:

- Specified the scope.
- Specified the contact option.
- Adapted restrictions and exclusions.

Specified the scope Since the organization is running the bug bounty program specifically for its e-health application, the scope was determined to include all assets of this product. Following the recommendations in section 5.1 of this thesis, it was determined that a specific scope will likely lead to better results. Therefore, the scope was specified as follows:

1. Domain and all subdomains of target application.
2. Android application.
3. iOS application.
4. All API endpoints contact by one of the mobile applications.

This scope definition includes an exact specification of the target's web and mobile applications, while having a catch-all entry for API endpoints. This entry allows API endpoints which are unknown to the organization, to be included in scope. Further, as defined in section 5.1.1, a statement which excludes all other targets from the scope of the bug bounty program has been adopted.

Specified the contact option For the purpose of this bug bounty program, the organization decided to use e-mail as the form of communication. As shown in section 5.1, this is also an approach suggested by this thesis. The e-mail address used for receiving vulnerability reports was decided to be *bugbounty@organziation-domain.com*. Additionally, the organization requested that PGP encryption is used for e-mail communication, and that it is made mandatory. In the example rules of engagement (section 5.1.4) of this thesis, this work suggests to use the formulation:

> *„To ensure confidentiality, we encourage you to encrypt all communication to us via e-mail.“*.

However, due to the high security requirements of the product, the organization decided to change it as follows:

> *„To ensure confidentiality, we **require** you to encrypt all communication to us via e-mail.“*.

Additionally, a statement was added to the rules of engagement which explains that unencrypted vulnerability reports will be considered a breach of this policy.

Finally, as suggested in section 5.1.1 of this thesis, a template for vulnerability reports has been provided. This template requires the following information to be included:

- Title
- Vulnerability type
- Short description of vulnerability and impact
- Affected product/service
- Detailed steps to reproduce
  - In case of web applications, including specific URI, parameter and a copy of the HTTP request(s)
  - For mobile applications, include all steps taken to decompile/reverse engineer the vulnerability
- A CVSS score
- The date and time of testing performed on the service/product

- The version of the application (if available)
- Proof of concept files

Adapted restrictions and exclusions As suggested in this thesis, the restrictions and exclusions given in the example rules of engagement have been adapted. The exclusions for web targets required no modifications, however, additional exclusions for android and iOS applications have been added. These are not provided as part of this thesis and were taken from other resources.

The testing restrictions as suggested in section 5.1.4 were also modified. Specifically, the organization decided to allow testing for denial of service attacks in the application layer, while keeping network-based denial of service attacks as out of scope. Additionally, automated scanning has been allowed. However, to avoid vulnerability reports taken directly from automated scanners, a statement which requires prior verification of these findings by the security tester has been added.

7.2.3 Creating the Internal Process and Appointing Stakeholders

Following the creation of rules of engagement, the internal processes for supporting the bug bounty program were defined. Here, the following items were discussed and agreed upon:

- Stakeholders and responsible personnel
- If and what software to use for processing vulnerability reports
- How reward amounts are decided and issued
- Creation of internal policy that documents the process

Stakeholders and responsible personnel For deciding the staff and its responsibilities, the stakeholders organized a meeting. The product owner and security team lead specified a project manager, a security lead and a bug bounty coordinator. Additionally, the person responsible for issuing rewards was appointed. It was also decided when the bug bounty program should go live, and what kind of marketing should be done around the launch. For this product, the decision was made to launch without marketing and observe interest before taking further steps in that direction.

If and what software to use for processing vulnerability reports Although the communication option was decided to be e-mail, for internal processing of vulnerability reports, the decision whether to use software such as a ticketing system had to be made. The stakeholders also assessed the impact of using a mail client, however, limitations in search and tracking assignments eliminated this option. Since the organization already had an internal ticketing system and it supports integration with an email address, the decision to use this software was made.

How reward amounts are decided and issued The stakeholders also decided the two activities required for rewarding valid vulnerabilities: determining the amount and issuing the reward. For determining the amount, the creation of a vulnerability reward panel was decided as suggested in section 5.1.2 of this thesis. The following members of the reward panel were appointed as mandatory participants:

1. Security lead
2. Penetration testing lead
3. Bug bounty coordinator

Additionally, mandatory attendance was requested for all security engineers who filed a vulnerability for rewarding. Optional attendance was decided for the project manager, product manager, and other personnel as needed. For managing the issuing of rewards, the bug bounty coordinator was appointed to send emails to the person responsible for issuing rewards. These emails include the contact details of the tester and the reward amount for transfer. Afterwards, the person responsible for issuing the rewards handles the process of requesting bank information and transferring the reward to the whitehat hacker.

Creation of internal policy that documents the process Finally, all these details decided by stakeholders were included in an internal policy document for reference. The document also includes details about the scope and rules of engagement, so readers are aware these exist and where to look for them.

7.2.4 Launching the Bug Bounty Program

As shown in chapter 6 of this thesis, some preparation is required before launching a bug bounty program. In section 6.1, it is recommended that success metrics are created and measured. A number of example metrics are also provided. For this case study, table 7.1 shows the selection of metrics their success criteria.

Metric	**Success value/range**
Total number of vulnerability reports	>100
Signal-to-noise ratio	At least 30% signal
Total amount of rewards	>10,000 EUR
Average amount of reward	500 EUR
Average first response	<3 days
Number of unique participants	50

Table 7.1: Success Metrics of Case Study

For target preparation as suggested in section 6.2.2, no preparation was required for this case study. Due to the complexity of the system, the organization decided to run the bug bounty program on their production environment. Additionally, it was decided that no testing credentials are created for the initial launch, and this decision would be revisited if participation is lower than desired.

The stakeholders also made some technical decisions regarding implementation. Due to this, certain resources were created to support the bug bounty program. These included setting up a ticketing system, creating the e-mail address for receiving vulnerabilities, creating pgp keys for encryption and publishing the rules of engagement. Finally, the bug bounty program was launched on the date decided by stakeholders. In practice, this action meant that the rules of engagement were made publicly available. As mentioned in the previous section, it was decided that no marketing efforts shall be taken for the launch. Internally, an email was sent to all stakeholders notifying them about the launch of the program.

7.2.5 Operating the Bug Bounty Program

Now that the bug bounty program was live, the organization was ready to receive vulnerability reports. Internally, the process for handling vulnerability reports worked as expected. The bug bounty coordinator and one security engineer were tasked with monitoring the queue and processing tickets. Vulnerabilities were triaged per the process shown in section 6.4.1 of this thesis. The ticketing system was used to assign outcomes to reports and send replies to researchers. For easier communication with whitehat hackers, the bug bounty coordinator created templated responses for similar vulnerability reports. These were taken and adopted in the internal policy document.

Because no valid vulnerabilities were reported, the reward process was not tested as part of this case study. Additionally, no requests for public disclosure were received. To increase interest and quality of vulnerability reports, stakeholders discussed about publicly marketing the bug bounty program. Publishing documentation regarding the target applications and their implementation was also considered. During the case study, these considerations were still being discussed and no actions was taken.

7.3 Expert Interviews

As part of this case study, expert interviews were conducted to evaluate the performance of the concepts suggested in this thesis. As shown by Bogner et al.[91], the use of expert interviews is a popular research methodology to validate design concepts, amongst other things.

For this case study, interviews with three IT Security experts were conducted. The interviews were carried out after the bug bounty program was launched, to allow reflecting back on the case study and analyse the results. In the following sections, the interview process and results are discussed. After the interview methodology is presented, the topics covered in the interviews are discussed. Finally, the results of the interviews are shown and analyzed.

7.3.1 Interview Methodology

For the purpose of evaluating the case study, this work uses a semi-structured interview methodology. As shown by Newcomer et al.[92], semi-structured interviews are a useful methodology to supplement and add depth to other research methods. A semi-structured interview allows to ask open-ended questions and obtain valuable information by participants. Through follow up questions and probing, formative evaluation of programs by interviewing key staff members can be achieved[92].

The concepts of this thesis are practically applied via the case study as shown in section 7.2. To support the case study, a semi-structured interview is considered to be an appropriate approach. The interview participants can give their opinion about the performance of this work applied in the case. By interviewing IT Security experts who have experience with offensive security testing, a qualitative evaluation of the case study can be performed. Additionally, the interviewer can ask follow-up questions and discuss potential improvements.

7.3.2 Interview Topics

The main focus of these expert interviews is to evaluate the performance of this thesis in the case study. Hence, most questions revolve around the implementation of the bug bounty program. Questions are asked to reason about the decision to run a bug bounty program and the expected results. Another topic covers the selection of the integration approach and reasons behind it. Additionally, a number of questions are asked to shed light on how the concepts of this thesis performed in creating rules of engagements, running and managing the bug bounty program.

A secondary goal of the interviews is to understand the experts' opinion on crowdsourced offensive security as a security measure in enterprise organizations. Questions regarding the expert's experience with bug bounty programs are asked, and their thoughts on the future of this methodology are discussed. The full list of questions can be found in the appendix of this thesis.

7.3.3 Results

The interviews began with introductions wherein the interviewees were asked about their current role and experience with offensive security. All interviewees hold positions relevant to offensive and defensive security, and have many years of experience in these fields. All interviewees also confirmed that they have experience with e-health products in some form or another. When asked about crowdsourced offensive security, only one interviewee had some passive experience with the field, without being actively involved. The other interviewees had no prior experience with crowdsourced offensive security.

The interview continued with questions whether offensive security and bug bounties are useful measures for e-health products. The interviewees all agreed that offensive security is a must-have measure for this type of product, because of the sensitive nature of the data. The interviewees also unanimously agreed that bug bounties provide their contribution in securing these products even if other offensive security measures have been applied. Two Interviewees mentioned the continuous testing aspect of bug bounties as an advantage compared to other measures which are performed for a limited time period. One interviewee stressed the need for controlling what testing is applied and when to stop. Further, two interviewees strongly advised that bug bounties cannot replace other offensive security measures and should not be considered an all-in-one solution to security testing.

For the interview questions particular to the case study, interviewees agreed that an institutional approach was the correct decision for this use case. The sensitive nature of the data involved, as well as having control over the program were the main reasons. Additionally, interviewees agreed that the rules of engagement template fulfils its job in providing a starting point for organizations wanting to launch a bug bounty program. Most comments and remarks were made about the roles suggestions of this work. All interviewees suggested that the bug bounty coordinator role can be replaced with a generic project manager. Additionally, interviewees remarked that software engineers would only access the result of crowdsourced offensive security, which is the reported vulnerabilities they would need to fix. Hence, they would not be directly involved with the bug bounty program.

After reviewing the case study, interviewees were asked whether their thoughts about crowdsourced offensive security have changed. All interviewees answered that their thoughts remained the same, and that crowdsourced offensive security is a valuable asset in offensive security efforts. When asked about the future of offensive security and bug bounties, one interviewee suggested the importance of both will drop. This was attributed to the prediction that programming will be largely automated in the future, and that hopefully results in less vulnerabilities.

7.4 Analysis

This section provides an analysis of the case study, and how the concepts of this thesis performed in practice. As shown in section 7.1, the bug bounty program is created for the e-health solution of an enterprise organization. Because an institutional bug bounty program was desired, the concepts presented in section 5.1 were used to drive the integration process. Additionally, the concepts in chapter 6 were used to launch and operate the bug bounty program, including vulnerability report processing and issuing rewards.

For creating the rules of engagement, the template in section 5.1.4 served as a good basis and required only small modifications. Building on top of such a template not only helped to speed up the process, but it gave stakeholders a clear picture of expectations. This in turn guided discussions around potential modifications and adaptions, making them practical and effective. One shortcoming of the template was noted regarding mobile applications. Specifically, the template does not provide a list of exclusions for android or ios apps. Nevertheless, by having web exclusions in the template, it was easy to create a similar list for mobile applications. Overall, the template was regarded as an important and effective item in the process of creating rules of engagement.

The conducted case study also followed the suggestion of section 5.1.2 in creating a bug bounty coordinator role. This proved to be very effective, as there was a single person in charge with the successful creation of the bug bounty program. Even though multiple stakeholders and staff members were involved and decisions had to be made, having the bug bounty coordinator role allowed for a single point of contact when it came to questions around the bug bounty program. The bug bounty coordinator also handled project management tasks such as organizing meetings and documentation, effectively removing the need for a dedicated project management person. In general, this case study showed that having a bug bounty coordinator role is advantageous, especially for organizations with little or no experience in crowdsourced offensive security.

As shown in section 6.1, success metrics are useful for measuring the performance of the bug bounty program. For this case study, only a subset of the given metrics were used, as shown in section 7.2.4. A selection of meaningful metrics had to be made, because of the characteristics of the target application. For example, due to the high security level of the system, only a small number of vulnerability reports were expected. Therefore, metrics regarding the incoming number of reports were considered to be inappropriate for determining the success of the program. Additionally, because the target was specified ahead of time, the selection process as shown in section 6.2.1 was not applied for this case study.

In preparation for launching the bug bounty program, questions regarding the handling of incoming vulnerability reports came up. While this work provides instructions for what the process looks like in section 4.1.3, there are no practical suggestions or software recommendations to manage resource allocation. One can argue that resource allocation is a generic problem and not specific to bug bounties[93], however, suggestions for practical solutions could steer organizations in the right direction. For organizations which already have effective resource allocation in place, such suggestions would offer a chance for comparison and adaptions as needed. In this case study, options for managing incoming vulnerability reports via email and a ticketing system were considered. Upon considerations regarding the designed bug bounty, the stakeholders decided to use a ticketing system.

Using a ticketing system provided a number of benefits in operating the bug bounty program. Besides resource allocation, the ticketing system enabled better search, tagging reports and abstracted encryption. Such functionality does not just make processing vulnerability reports easier, but it also makes the process faster. In situations when a large number of vulnerabilities are reported in a short amount of time, this can be a key factor for meeting success criteria and maintaining a healthy program. Contributing factors also appeared to be the experience of security engineers with ticketing systems, as well as their ease of use. Providing regular trainings for the ticketing system functionality and bug bounty process may be effective, especially when there are rotations of security engineers doing triage work.

Expert interviews provided valuable insight and perspectives from people with different roles and viewpoints. For example, the interviews showed that roles required for running a bug bounty program are not static and should be adapted depending on the organization's human and IT Security capacities. Additionally, the different views on whitehat hackers coming in contact with production data was an interesting aspect. While one interviewee suggested this can result in a massive leak, another saw this as a positive situation when bug bounty participants find these vulnerabilities instead of malicious attackers. Both viewpoints are warranted and potential situations that may arise. The actual outcome likely depends on a number of factors, including the researcher ethics, the bug bounty program reputation and maximum reward amounts.

The expert interviews also confirmed the pre-adoption fears mentioned in the problem statement of this work. While crowdsourced offensive security is not an unknown term, the idea of inviting external people to perform offensive security testing seems dangerous. A lot of interviewee remarks and comments are made around control and protection from undesired situations. Such situations could be when production data are accessed, coming in contact with legitimate users or disturbing legitimate functioning of the targets being tested. Of course the same can happen with other offensive security methods such as penetration testing, however, the existence of a contract and closer contact with the testers appears to this reduce this concern here. As a result, the concept of bug bounties must be marketed and sold internally. This includes guides and operational instructions for avoiding undesired situations, but taking examples from other organizations will likely contribute as well. Especially if organizations in the same business spectrum are already using crowdsourced offensive security successfully, case studies and testimonials may ease some of the fears of using this offensive security measure.

Finally, the existence of a bug bounty program provides a good incentive for doing the right thing when a vulnerability has been discovered. As mentioned by interviewees, malicious attackers will always be present and attempt to penetrate systems. However, the existence of a bug bounty program not only drives more security testing from individuals looking to make money, it also serves as a communication vehicle for those that stumble upon a vulnerability and want to report it with ethical obligation. In the end, there is no such thing as absolute security and crowdsourced offensive security is another layer of prevention in the security efforts of organizations.

8 Conclusion and Further Work

The principal goal of this thesis is to make crowdsourced offensive security approachable in enterprise organizations. The integration approach in chapter 5 and operational guidelines in chapter 6 provide practical instructions for creating and managing a bug bounty program. To evaluate this work, a case study has been conducted. Additionally, IT Security experts have been interviewed to assess the concepts of this thesis and how they performed in the case study. In summary, this thesis gives answers to the following research questions:

- How can enterprise organizations integrate a bug bounty program as an offensive security measure?
- Is crowdsourced offensive security an effective measure to improve the security posture of an enterprise organization?
- Can bug bounties complement penetration tests?

The analysis of security standards in chapter 3 of this thesis showed that despite its rising popularity, crowdsourced offensive security has not been adapted in existing resources for enterprise IT Security. The only standard that covered some aspects of bug bounties was ISO 29147[65]. However, ISO 29147 only overlaps with this work in the area of handling vulnerability reports from external researchers. This thesis finds that there is a lack of comprehensive information in acknowledged security guidelines for enterprise organizations wanting to use crowdsourced offensive security. With its rising popularity and demonstrated effectiveness, it can be concluded that security standards need to adopt this emerging offensive security method.

To answer the first research question, chapter 5 of this thesis offers insight into the two most common ways to integrate crowdsourced offensive security testing: an institutional approach and using a bug bounty services provider. It was shown that each approach has its advantages and may be useful for certain situations. This thesis concludes that these two integration options offer good alternatives for different types of organizations and their requirements. The institutional approach presents an attractive option for organizations wanting full control. However, the case study shows that having the necessary resources to create and manage the bug bounty program is a key factor for its successful execution. For organizations without such resources, using a bug bounty services provider and one of its time-based offerings will likely result in a more cost effective solution.

For the third research question, chapter 6 of this thesis provides practical guidelines for preparing, launching and managing a bug bounty program in an institutional approach. The effectiveness of crowdsourced offensive security, and its place in the IT Security program of an organization was also discussed. As a result of this discussion, this thesis finds that crowdsourced security testing does not replace other offensive security methods. Instead, it provides the most value when applied as a complementary measure in addition to traditional offensive security methods such as penetration testing and red-teaming. For example, while bug bounties excel with low latency vulnerability detection and a large number of skilled testers, they do not guarantee methodology and coverage. On the other hand, penetration testing guarantees exactly these items by following a standardized method of testing and covering the entire application. Together, these two offensive security methods offer the best of both. This thesis also concludes that offensive security is an effective measure to improve the security of enterprise organizations, thereby answering the second research question. Alongside the benefits of low latency vulnerability detection and faster remediation as shown in chapter 6, bug bounties provide talented hackers an ethical and legal way to earn money. In the past, the only way to profit from a vulnerability discovery has been to sell it in the black market[94]. With bug bounties offering lucrative monetary rewards, an incentive to ethically report the vulnerability to the vendor has been created. Hence, it can be concluded that crowdsourced offensive security provides many contributions to the improvement of an organization's security posture.

As shown in the problem statement of this work, although crowdsourced offensive security has been around for long it only gained popularity in recent years. Hence, this work recognizes that this field is dynamic and continuously evolving, and that running bug bounty programs may change in the future. This work also recognizes its limited testing through the single case study of chapter 7. Future work may apply the concepts in this thesis to various organizations of different branches and evaluate its performance. Undoubtedly, there will be room for improvement, some of which have already been identified in section 7.4. Nevertheless, this thesis answered important questions about the usefulness of crowdsourced offensive security, while at the same time providing a clear guide for enterprise organizations wanting to get started with it.

This work has been created with the purpose of finding application in any large organization wanting to adopt this emerging offensive security method. It is mainly aimed as a practical resource for quickly integrating and using bug bounty programs. As a secondary purpose, this work aims to fill the gap of information in current security standards and guidelines. This thesis also concludes that more research is needed in the integration process for crowdsourced offensive security. The literature research conducted as part of this work shows that this field is actively being researched, however, most papers are centered around different aspects such as crowd incentives or empirical analysis. Research around integration concepts and operational aspects have only been found sparsely. This work also finds that significant research contributions were made by bug bounty services providers, with annual reports about statistics and bug bounty adoption. However, further research is needed to investigate the learnings and processes that these providers use for running and managing programs. This may present a challenging endeavour since such information may be considered intellectual properly. Researching organizations with institutional bug bounty programs may also present a challenge for the same reasons. Nevertheless, this thesis finds that the contributions of crowdsourced offensive security are significant enough for investing resources and research time in this dynamic new field.

Bibliography

References

[1] C. Easttom, *Computer Security Fundamentals*, Pearson, 2016, ISBN: 978-0-7897-5746-3.

[3] P. E. Proctor and C. Byrnes, *The Secured Enterprise: Protecting Your Information Assets*, Prentice Hall PTR, 2002, ISBN: 978-0-13-061906-8.

[4] P. Kim, *The Hacker Playbook 3: Practical Guide to Penetration Testing*, Independently published, 2018, ISBN: 978-1-980901-75-4.

[5] N. Munaiah and A. Meneely, *Vulnerability severity scoring and bounties: Why the disconnect?* ACM Press, 2016. DOI: 10.1145/2989238.2989239.

[6] T. Maillart *et al.*, *Given enough eyeballs, all bugs are shallow? Revisiting Eric Raymond with bug bounty programs*, in „Journal of Cybersecurity“ 2017. DOI: 10.1093/cybsec/tyx008.

[7] A. Magazinius *et al.*, *Bug Bounty Programs – A Mapping Study* 2019. DOI: 10.1109/SEAA.2019.00070.

[8] M. Al-Banna *et al.*, *Friendly Hackers to the Rescue: How Organizations Perceive Crowdsourced Vulnerability Discovery* 2018. [Online]. Available: https://aisel.aisnet.org/pacis2018/230/ (last visited on 02/16/2021).

[10] H. Fryer and E. Simperl, *Web Science Challenges in Researching Bug Bounties* ACM Press, 2017. DOI: 10.1145/3091478.3091517.

[11] J. Ruohonen and L. Allodi, *A Bug Bounty Perspective on the Disclosure of Web Vulnerabilities* 2018. arXiv: 1805.09850.

[12] M. Zhao *et al.*, *An Empirical Study of Web Vulnerability Discovery Ecosystems* ACM Press, 2015. DOI: 10.1145/2810103.2813704.

[13] M. Zhao *et al.*, *Crowdsourced Security Vulnerability Discovery: Modeling and Organizing Bug-Bounty Programs*, in „Proc. of the 4th AAAI Workshop on Mathematical Foundations of Human Computation“ 2016. [Online]. Available: http://aronlaszka.com/papers/zhao2016crowdsourced.pdf (last visited on 02/16/2021).

[14] S. S. Malladi and H. C. Subramanian, *Bug Bounty Programs for Cybersecurity: Practices, Issues, and Recommendations*, in „IEEE Software“ 2019. DOI: 10.1109/MS.2018.2880508.

[15] M. Zhao *et al.*, *Devising Effective Policies for Bug-Bounty Platforms and Security Vulnerability Discovery*, in „Journal of Information Policy“ 2017. DOI: 10.5325/jinfopoli.7.2017.0372.

[16] D. Luna *et al.*, *Productivity and Patterns of Activity in Bug Bounty Programs: Analysis of HackerOne and Google Vulnerability Research* ACM Press, 2019. DOI: 10.1145/3339252.3341495.

[17] A. Kuehn and M. Mueller, *Analyzing Bug Bounty Programs: An Institutional Perspective on the Economics of Software Vulnerabilities*, in „SSRN Electronic Journal“ 2014. DOI: 10.2139/ssrn.2418812.

[18] J. M. Anderson, *Why we need a new definition of information security*, in „Computers & Security“ May 2003. DOI: 10.1016/S0167-4048(03)00407-3.

[19] S. Bosworth *et al.*, *Computer Security Handbook*, Wiley, 2014, ISBN: 978-1-118-12706-3.

[20] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley, 2003, ISBN: 978-0-201-44099-7.

[21] D. Mellado and D. G. Rosado, *An Overview of Current Information Systems Security Challenges and Innovations J.UCS Special Issue*, in „Journal of Universal Computer Science“. [Online]. Available: http://www.jucs.org/jucs_18_12/an_overview_of_current/jucs_18_12_1598_1607_editorial.pdf (last visited on 11/20/2020).

[22] D. Dzung *et al.*, *Security for Industrial Communication Systems*, in „Proceedings of the IEEE“ Jun. 2005. DOI: 10.1109/JPROC.2005.849714.

[23] S. Widup *et al.*, *2015 Verizon Data Breach Investigations Report* 2015. DOI: 10.13140/RG.2.1.4205.5768.

[24] P. Rosati *et al.*, *Social media and stock price reaction to data breach announcements: Evidence from US listed companies*, in „Research in International Business and Finance“ Jan. 2019. DOI: 10.1016/j.ribaf.2018.09.007.

[26] C. M. Hayes, *Comparative Analysis of Data Breach Laws: Comprehension, Interpretation, and External Sources of Legislative Text*, in „SSRN Electronic Journal“ 2019. DOI: 10.2139/ssrn.3334688.

[27] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, Pearson, 2015, ISBN: 978-0-13-377392-7.

[28] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, Prentice Hall, 2007, ISBN: 978-0-13-239077-4.

[29] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Wiley, 1996, ISBN: 978-0-471-12845-8.

[30] J. Andress, *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*, Elsevier/Syngress, 2014, ISBN: 978-0-12-800744-0.

[31] E. Kolodenker *et al.*, *PayBreak: Defense Against Cryptographic Ransomware* ACM Press, 2017. DOI: 10.1145/3052973.3053035.

[32] J.-P. Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*, No Starch Press, 2017, ISBN: 978-1-59327-882-3.

[33] A. Tripathi and U. K. Singh, *Towards Standardization of Vulnerability Taxonomy* IEEE, Nov. 2010. DOI: 10.1109/ICCTD.2010.5645826.

[34] D. R. Kuhn *et al.*, *An Analysis of Vulnerability Trends, 2008-2016* IEEE, Jul. 2017. DOI: 10.1109/QRS-C.2017.106.

[35] A. Maurushat, *Disclosure of Security Vulnerabilities*, Springer London, 2013, ISBN: 978-1-4471-5003-9.

[36] European Union and Agency for Network and Information Security, *ENISA Threat Landscape Report 2018: 15 Top Cyberthreats and Trends.* 2019, ISBN: 978-92-9204-286-8.

[37] D. Mitropoulos *et al.*, *Defending Against Web Application Attacks: Approaches, Challenges and Implications*, in „IEEE Transactions on Dependable and Secure Computing“ Mar. 1, 2019. DOI: 10.1109/TDSC.2017.2665620.

[38] Y. Diogenes and E. Ozkaya, *Cybersecurity - Attack and Defense Strategies - Second Edition*, 2019, ISBN: 978-1-83882-779-3.

[39] M. Abliz, *Internet Denial of Service Attacks and Defense Mechanisms*. [Online]. Available: https://blog.oureducation.in/wp-content/uploads/2014/06/Internet-Deniel.pdf (last visited on 11/20/2020).

[40] P. Shedden *et al., Asset Identification in Information Security Risk Assessment: A Business Practice Approach*, in „Communications of the Association for Information Systems“ 2016. DOI: 10.17705/1CAIS.03915.

[41] S. E. Donaldson *et al.*, *Enterprise Cybersecurity Study Guide*, Apress, 2018, ISBN: 978-1-4842-3257-6.

[42] Joint Task Force Transformation Initiative, *„Security and Privacy Controls for Federal Information Systems and Organizations“* 2013. DOI: 10.6028/NIST.SP.800-53r4.

[43] I. Arce and G. McGraw, *Guest Editors' Introduction: Why Attacking Systems Is a Good Idea*, in „IEEE Security and Privacy Magazine“ Jul. 2004. DOI: 10.1109/MSP.2004.46.

[44] Y. Stefinko *et al., Manual and Automated Penetration Testing. Benefits and Drawbacks. Modern Tendency* IEEE, Feb. 2016. DOI: 10.1109/TCSET.2016.7452095.

[45] K. Xynos and I. Sutherland, *Penetration Testing and Vulnerability Assessments: A Professional Approach* 2010. [Online]. Available: https://ro.ecu.edu.au/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=1015&context=icr (last visited on 02/26/2021).

[46] J. G. Oakley, *Professional Red Teaming: Conducting Successful Cybersecurity Engagements*, Apress, 2019, ISBN: 978-1-4842-4308-4.

[47] G. Weidman, *Penetration Testing: A Hands-on Introduction to Hacking*, No Starch Press, 2014, ISBN: 978-1-59327-564-8.

[48] S.-P. Oriyano, *Penetration Testing Essentials*, John Wiley & Sons, Inc., 2018, ISBN: 978-1-119-23530-9.

[50] R. Pompon, *IT Security Risk Control Management: An Audit Preparation Plan*, Apress, 2016, ISBN: 978-1-4842-2140-2.

[51] J. Rehberger, *Cybersecurity Attacks - Red Team Strategies: A Practical Guide to Building a Pentest Program Having Homefield Advantage.* Packt Publishing Limited, 2020, ISBN: 978-1-83882-886-8.

[52] H. Susanto *et al., Information Security Management System Standards: A Comparative Study of the Big Five* 2011. [Online]. Available: https://www.academia.edu/download/30294093/113505-6969-ijecs-ijens.pdf (last visited on 11/20/2020).

[53] K. Haufe *et al., Security Management Standards: A Mapping*, in „Procedia Computer Science“ 2016. DOI: 10.1016/j.procs.2016.09.221.

[56] W. Tounsi and H. Rais, *A survey on technical threat intelligence in the age of sophisticated cyber attacks*, in „Computers & Security“ Jan. 2018. DOI: 10.1016/j.cose.2017.09.001.

[57] O. Santos, *Developing Cybersecurity Programs and Policies*, Pearson, 2019, ISBN: 978-0-7897-5940-5.

[58] K. J. Knapp *et al., Information security policy: An organizational-level process model*, in „Computers & Security“ Oct. 2009. DOI: 10.1016/j.cose.2009.07.001.

[59] A. Kohnke *et al.*, *The Complete Guide to Cybersecurity Risks and Controls*, 2016, ISBN: 978-1-4987-4057-9.

[60] I. Yevseyeva *et al., Selecting Optimal Subset of Security Controls*, in „Procedia Computer Science“ 2015. DOI: 10.1016/j.procs.2015.08.625.

[61] L. Almeida and A. Respício, *Decision support for selecting information security controls*, in „Journal of Decision Systems“ May 15, 2018. DOI: 10.1080/12460125.2018.1468177.

[62] J. Breier and L. Hudec, *On Selecting Critical Security Controls* IEEE, Sep. 2013. DOI: 10.1109/ARES.2013.77.

[63] T. Schreider, *Building an Effective Cybersecurity Program*, Rothstein Publishing, 2019, ISBN: 978-1-944480-53-0.

[64] C. J. P. Moschovitis, *Cybersecurity Program Development for Business: The Essential Planning Guide*, Wiley, 2018, ISBN: 978-1-119-43000-1.

[66] J. H. P. Eloff and M. Eloff, *Information Security Management: A New Paradigm*, ser. SAIC-SIT ’03 South African Institute for Computer Scientists and Information Technologists, 2003, ISBN: 1-58113-774-5.

[67] E. Humphreys, *Information security management system standards*, in „Datenschutz und Datensicherheit - DuD“ Jan. 2011. DOI: 10.1007/s11623-011-0004-3.

[68] K. Dempsey *et al.*, *„Summary of NIST SP 800-53 Revision 4, Security and Privacy Controls for Federal Information Systems and Organizations“* Feb. 2014. DOI: 10.6028/NIST.CSWP.02192014.

[69] T. Walshe and A. Simpson, *An Empirical Study of Bug Bounty Programs* IEEE, Feb. 2020. DOI: 10.1109/IBF50092.2020.9034828.

[72] L. Bell *et al.*, *Agile Application Security: Enabling Security in a Continuous Delivery Pipeline*, O’Reilly Media, 2017, ISBN: 978-1-4919-3884-3.

[73] A. Laszka *et al.*, *The Rules of Engagement for Bug Bounty Programs*, ser. Lecture Notes in Computer Science Springer Berlin Heidelberg, 2018. DOI: 10.1007/978-3-662-58387-6_8.

[75] A. Laszka *et al.*, *Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms*, ser. Lecture Notes in Computer Science Springer International Publishing, 2016. DOI: 10.1007/978-3-319-45741-3_9.

[76] S. Patil *et al.*, *Design of Efficient Web Vulnerability Scanner* IEEE, Aug. 2016, ISBN: 978-1-5090-1285-5. DOI: 10.1109/INVENTIVE.2016.7824873.

[77] T. Zimmermann *et al.*, *What Makes a Good Bug Report?*, in „IEEE Transactions on Software Engineering“ Sep. 2010. DOI: 10.1109/TSE.2010.63.

[78] J. Großmann and F. Seehusen, *Combining Security Risk Assessment and Security Testing Based on Standards*, ser. Lecture Notes in Computer Science Springer International Publishing, 2015. DOI: 10.1007/978-3-319-26416-5_2.

[80] M. Weulen Kranenbarg *et al.*, *Don’t shoot the messenger! A criminological and computer science perspective on coordinated vulnerability disclosure*, in „Crime Science“ Dec. 2018. DOI: 10.1186/s40163-018-0090-8.

[81] S. L. Garfinkel *et al.*, *How to make secure email easier to use* ACM Press, 2005. DOI: 10.1145/1054972.1055069.

[82] J. Erbes *et al.*, *The Future of Enterprise IT in the Cloud*, in „Computer“ May 2012. DOI: 10.1109/MC.2012.73.

[83] Y. Makino and V. Klyuev, *Evaluation of Web Vulnerability Scanners* IEEE, Sep. 2015. DOI: 10.1109/IDAACS.2015.7340766.

[84] C. Hadnagy, *Social Engineering: The Art of Human Hacking*, Wiley, 2011, ISBN: 978-0-470-63953-5.

[85] S. F. Dizaji and M. R. Farzanegan, *„Do sanctions reduce the military spending in Iran?“* 2018. [Online]. Available: https://www.econstor.eu/handle/10419/200687 (last visited on 11/20/2020).

[86] N. de Guzman Chorny *et al.*, *The state of child labor protections in 193 countries: Are countries living up to their international commitments?*, in „International Journal of Sociology and Social Policy“ Aug. 22, 2019. DOI: 10.1108/IJSSP-12-2018-0229.

[87] A. T. Chatfield and C. G. Reddick, *Cybersecurity Innovation in Government: A Case Study of U.S. Pentagon's Vulnerability Reward Program* ACM, Jun. 7, 2017. DOI: 10.1145/3085228.3085233.

[88] J. F. Ransome *et al.*, *Core Software Security: Security at the Source*, CRC Press, an Auerbach book, 2014, ISBN: 978-1-4665-6096-3 978-1-4665-6095-6.

[89] C. Schanes *et al.*, *Problem Space and Special Characteristics of Security Testing in Live and Operational Environments of Large Systems Exemplified by a Nationwide IT Infrastructure* IEEE, Sep. 2009. DOI: 10.1109/VALID.2009.24.

[90] A. Poller *et al.*, *Can Security Become a Routine?: A Study of Organizational Change in an Agile Software Development Group* ACM, Feb. 25, 2017. DOI: 10.1145/2998181.2998191.

[91] A. Bogner *et al.*, *Introduction: Expert Interviews — An Introduction to a New Methodological Debate* Palgrave Macmillan UK, 2009. DOI: 10.1057/9780230244276_1.

[92] K. E. Newcomer *et al.*, *Handbook of Practical Program Evaluation*, Jossey-Bass & Pfeiffer Imprints, Wiley, 2015, ISBN: 978-1-118-89361-6 978-1-118-89369-2.

[93] S. Bouajaja and N. Dridi, *A survey on human resource allocation problem and its applications*, in „Operational Research“ Jul. 2017. DOI: 10.1007/s12351-016-0247-8.

[94] J. Radianti and J. Gonzalez, *Understanding Hidden Information Security Threats: The Vulnerability Black Market* IEEE, 2007. DOI: 10.1109/HICSS.2007.583.

Online References

[2] 2019-Data-Breach-Investigations-Report.Pdf, [Online]. Available: https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf (last visited on 05/24/2021).

[9] HackerOne. The Hacker-Powered Security Report 2019, [Online]. Available: https://www.hackerone.com/resources/reporting/the-hacker-powered-security-report-2019 (last visited on 05/24/2021).

[25] 2019 Cost of a Data Breach Report, [Online]. Available: https://databreachcalculator.mybluemix.net/executive-summary (last visited on 05/24/2021).

[49] ISO/IEC 27001 Information security management, ISO, [Online]. Available: https://www.iso.org/isoiec-27001-information-security.html (last visited on 05/24/2021).

[54] TIBER-EU FRAMEWORK – How to implement the European framework for Threat Intelligence-based Ethical Red Teaming, [Online]. Available: https://www.ecb.europa.eu/paym/cyber-resilience/tiber-eu/html/index.en.html (last visited on 05/24/2021).

[55] Österreichisches Informationssicherheitshandbuch, [Online]. Available: https://www.sicherheitshandbuch.gv.at/index.php (last visited on 05/24/2021).

[65] ISO/IEC 29147:2018, ISO, [Online]. Available: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/23/72311.html (last visited on 05/24/2021).

[70] Bugcrowd. Bugcrowd's Priority One Report 2019, Bugcrowd, [Online]. Available: https://static.carahsoft.com/concrete/files/2215/7296/5388/Bugcrowd_Priority_One_Report_2019.pdf (last visited on 05/24/2021).

[71] Bugcrowd. 2020 Inside the Mind of a Hacker Report, [Online]. Available: https://itmoah.bugcrowd.com (last visited on 05/24/2021).

[74] 2018 State of Bug Bounty, Bugcrowd, [Online]. Available: https://www.bugcrowd.com/resources/reports/state-of-bug-bounty-2018 (last visited on 05/24/2021).

[79] HackerOne. The 2020 Hacker Report, [Online]. Available: https://www.hackerone.com/resources/reporting/the-2020-hacker-report (last visited on 05/24/2021).

A Appendix

A.1 Expert Interviews

This section describes the interview process and lists the questions used during the expert interviews performed in section 7.3 of this thesis. The interview is structured with categories. Each category holds a number of questions, which are asked after a brief introduction to the category topic.

Category 1 Introductions.
Introduction: Interviewer introduces himself and presents the approach, course and outcomes of the case study. Then the following questions are asked:

1. What is your current role and experience?
2. Do you have experience with offensive security and bug bounties?
3. Do you have experience with e-health products?
4. What is your opinion in using offensive security for e-health products?

Category 2 General questions about the case study.
Introduction: Interviewer presents the case study and decision to run a bug bounty program. Then the following questions are asked:

1. Do you think crowdsourced offensive security is a useful measure for this product?
2. Can bug bounties contribute if other offensive security measures have been applied?
3. What challenges or problems do you anticipate for running a bug bounty program against this product?

Category 3 Integration approach.
Introduction: Interviewer presents the integration options provided by this thesis, and the chosen approach for the case study. Interviewer also shows the example rules of engagement provided in this thesis. Then the following questions are asked:

1. Do you think choosing an institutional bug bounty program was the correct approach for this use case?
2. Would you consider using a services provider for this type of product?
3. Are the rules of engagement clear and easy to understand?
4. Is it clear which targets and actions are in-scope for this bug bounty program?
5. Is the process for reporting vulnerabilities clear and easy to understand?

Category 4 Project architecture.
Introduction: Interviewer presents how the concepts of this thesis are applied in creating the internal processes and roles required for running the bug bounty program. Then the following questions are asked:

1. Are the roles clear and easy to understand?
2. Do you consider the bug bounty coordinator role necessary?
3. Do you see anything missing or improvement opportunities?

Category 5 Operational processes.
Introduction: Interviewer presents how the concepts of this thesis are applied in operating the bug bounty program. Then the following questions are asked:

1. Is the processing of vulnerability reports clear and easy to understand?
2. Are the instructions sufficient for getting started if a security engineer has had no prior experience with bug bounty programs?
3. Do you consider using a ticketing system to be the right approach? What other software could be used for managing the workload?
4. Would you prefer to reward after an issue has been fixed or does it suffice to approve it?

Category 6 Conclusion.
Introduction: Interviewer concludes the interview with the following questions:

1. Is there anything missing that you would consider adding to this work?
2. After having gone through the use case, what are your thoughts on crowdsourced offensive security as a security measure?
3. How do you expect offensive security to evolve in the future, and how do bug bounties fit in this picture?