A FRAMEWORK TO ASSESS BUG-BOUNTY PLATFORMS BASED ON POTENTIAL ATTACK VECTORS

by

Susan Ann McCartney

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

December 2022

DEDICATION

I dedicate this to my grandma Nancy, who inspired me to explore programs on the computer at a young age and go into cybersecurity. She was also an inspiration by getting a degree in computer science and programming in the early languages of Pascal, FORTRAN, and Assembly.

I also dedicate this to my current fiancee and future husband, Spencer, who presented me with the opportunity to program and helped me with classwork during the COVID-19 pandemic. He's also pushed me to learn, grow, and stretch my knowledge by challenging my ideas, from that I discovered the drive and determination I had to finish this paper.

I dedicate this to my parents, Tracy and Sandy, for always supporting me, emotionally and financially, and ensuring that I received all the educational opportunities presented to me.

ACKNOWLEDGEMENTS

I would like acknowledge Chris Perkins and Jason Herbst from Medtronic for the information on bug-bounty programs and platforms from a vendor's perspective. Alex Stevenson from Workiva for acting as the middle man to receive answers from BugCrowd about the bug-bounty process. Reese Pearsall from Montana State University for assisting in writing the malware sample script. Hoplite Industries for providing the malware sample artifacts. The Department of Homeland Security for funding my research. Finally, I would like to acknowledge Dr. Clemente Izurieta and Dr. Derek Reimanis for guiding me through the research process and providing feedback on the design of my conceptual framework.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

De nition 1.2.2: Bug-Bounty Platform
A third-party service that maintains a web interface for companies that wish to host a bug bounty program.

De nition 1.2.1: Bug-Bounty Program
A strategy used by companies who wish to improve the security of their software or products to deter hackers from exploiting their security vulnerabilities. There is a monetary benefit for reporting security vulnerabilities.

De nition 7.0.1: Conceptual Framework
A framework that allows one to generate a theory based on the combination of other existing theories or phenomena. A conceptual framework is used to explain the behavior when all the components interact with each other.

De nition 3.2.1: Framework
A set of guidelines or requirements that a company should follow.

De nition 2.1.0: Hacker
A person who enjoys the intellectual challenge of over creatively overcoming limitations.

De nition 2.5.1: Malware
A script or file that is used to disrupt activity of cause mischief to the user.

De nition 1.2.6: Penetration Testing
Security testing used to identify any holes in the software system that could cause a vulnerability exploitation.

De nition 1.2.3: Researcher
A white hat hacker who reports security vulnerabilities to the bug bounty program.

De nition 1.2.7: Red Teaming
The offensive security team that does penetration testing internally in the company.

De nition 1.2.4: Vendor
A company that hosts a bug bounty program on the platform.

De nition 1.2.5: Vulnerability Disclosure Policy
A policy that defines what vulnerabilities are considered valid based on scope and format.

ABSTRACT

Corporate computer security is becoming increasingly important because the frequency and severity of cyberattacks on businesses is high and increasing. One way to improve the security of company software is for a company to hire a third party to identify and report vulnerabilities, blocks of code that can be exploited. A bug-bounty program incentivizes ethical hackers (herein, 'researchers') to find and fix vulnerabilities before they can be exploited. For this reason, bug-bounty programs have been increasing in popularity since their inception a decade ago. However, the increase in their use and popularity also increases the likelihood of the companies being targeted by malicious actors by using a bug-bounty programs as the medium.

The literature review and investigation into the rules and requirements for bug-bounty platform revealed that though the bug-bounty programs can improve a vendor's security, the programs still contain a serious security flaw. The platforms are not required to scan reports for malware and there is no guidance requesting the vendors scan for malware. This means it is possible to perform a cyberattack using malware as a report attachment.

Through data collection from 22 platforms, an observational case study, and analysis of different malware, I have created a tool to assist vendors in selecting the platform of best fit and characterize the possible attack surfaces presented from the file options allowed on the platform. The outcome from this research is evidence of the importance of understanding the malware files used as report attachments. However, more research is needed in the relationship between file extensions and malware in order to thoroughly comprehend the attack surface capabilities, and to understand the trade-offs between security and convenience.

1

INTRODUCTION

Cybersecurity is becoming highly important because the frequency and severity of attacks is increasing [40]. The popularity of exploiting vulnerabilities is increasing.[1] Exploits and public disclosures of vulnerabilities have damaged companies' reputations or, in some cases, have caused companies to go out of business [26, 28].

In the past decade, the development of bug-bounty platforms has been used to help improve security for practitioners. The platforms provide a cost alternative way for vendors and practitioners to balance in-house and third-party security. The goal of a bug-bounty program is to help the vendor find vulnerabilities and fix/patch them before the vulnerabilities can be exploited. A **Bug-Bounty Program** is a strategy used by companies who wish to improve the security of their software or products to deter hackers from exploiting their security vulnerabilities. There is a monetary benefit for reporting security vulnerabilities.

The use of a bug-bounty program benefits both hackers and companies. Bug-bounty programs provide hackers with a legal way to find vulnerabilities and provide an incentive to disclose their findings to the company. The benefit to a company is to allow them time to fix the reported vulnerabilities with assurance that the vulnerability is not publicly disclosed until the issue is resolved.

More companies are hosting a bug-bounty program on a platform to improve their cybersecurity in an effort to maintain confidentiality of their customer's data. A bug-bounty platform is used to inform vendors of bugs in their software, leaving time for vendors to focus on other company obligations. However, there is a security flaw with the platform: there is

[1] CVE count by year

no requirement for protection or guidance for practitioners responsible for interpreting the results from the bug-bounty program. This means there is no way to avoid successful attacks through the vulnerability reports.

Therefore, the objective of this research is to create a tool for vendors to assist in selecting the bug-bounty platform that best fits their needs. In addition, I want to assist vendors in characterizing the landscape of attack surfaces through malware analysis and an observational case study of a set of malware because malware types are platform dependent.

This paper is an illustration of the important concepts required for the tool, an outline of the process for collected data and the results from the case study, and an analysis of the components of the tool. Chapter Two defines the important concepts of the tool. Chapter Three defines the purpose for each concept. Chapter Four describes the research goals and the questions that align with the structure of the conceptual framework. Chapter Five describes the data collection process using abduction and induction approaches. Chapter Six outlines the observational case study, analyzes and determines the meaning of the collected data, and determines how to portray the results from our research. Chapter Seven illustrates the tool and demonstrates its use cases for assessing the potential attack surfaces. Finally, Chapter Eight forms the conclusion of my findings and areas for future research.

3

BACKGROUND

There are three components to this framework. All have been thoroughly researched, but these concepts have not been connected. This chapter creates the connections between bug-bounty, files, and malware to help produce my final tool. There are several terms that need illustration before I can identify and define the concepts.

Hackers

According to Ellis's article on bug bounties [9], a **hacker** is defined as a person who enjoys the intellectual challenge of creatively overcoming limitations. An ethical hacker or (herein **researcher**) is the main type of hacker in this research. Researchers abide by the law and obtain permission to hack, reporting their findings to the company - software owner. The use of researchers provides an innovative, agile, and low-cost alternative to companies, especially in relation to bug-bounty programs [9].

Not all hackers are good, some hackers have malicious intent and want to destroy companies - to steal data and sell it for profit [13, 14]. The goal for these hackers, or black hat hackers, is to disrupt availability, violate confidentiality, and compromise integrity. This is accomplished by exploiting vulnerabilities. A **vulnerability** is a block of code or a piece of software that is susceptible to unauthorized access.

There are companies in the cyber world that use the detection of vulnerabilities to make a profit - the bug-bounty platforms, which connect researchers and companies to improve corporate security for software.

Bug-Bounty

There are two types of bug-bounty platforms: in-house and third-party. The scope of this research is on third-party platforms. The main purpose of the platform is to provide

services to companies and a channel of communication for researchers. Most of the services provided for the bug-bounty programs is to validate[1] and triage[2] reported vulnerabilities. Once the reports have been validated and triaged the vendor is notified that the reports are ready for viewing. A **vendor** is the term used to label a company that hosts a bug-bounty program on a platform.

In-House Platform

An in-house bug-bounty platform is handled internally by the company's employees. In order for companies to manage an in-house platform, the company needs to have the resources to manage a bug-bounty program. These duties include validating, triaging, and remediating all vulnerability submissions. Google, Microsoft, and Facebook all host their own in-house platform, but use a third-party platform to handle the bounty payouts and pool of researchers.[3]

Third-Party Platforms

A third-party **platform** is a company that provides software as a service for companies that want to improve their security at minimal cost and time [17]. Platforms have a range of services available to vendors, from acting as the middle man between researcher and vendor to filtering and sorting vulnerability submissions [24]. In exchange for use of the platform's services, the platform charges a fee for each bounty payout [3]. The third-party platform is mainly used for filtering submissions by removing invalid or duplicate entries. These types of submission make up a significant amount of all entries [20, 37, 38].

There are many bug-bounty programs that require researchers' participation in order to thrive and have a successful security program. The bug-bounty program gets its name

[1]Remove reports that are not correctly formatted, are not in scope of the policy, have already been reported, and have no relevance to the companies security.

[2]Sort reports based on severity of reported vulnerability and level of impact of software availability.

[3]https://bughunters.google.com/Google bug-bounty

because if a vulnerability is considered 'valid,' then the researcher receives a reward for reporting the vulnerability. But rewards are not restricted to monetary benefits. Points and swag can also be a reward [3, 18, 24]. A bug-bounty program is successful for a company when there is a balance between utilizing in-house protection (i.e. security team) and the use of a bug-bounty program [39]. Some company program options include run private, public, and a mixture of public and private.

As a middle man, the platform ensures the researcher abides with the disclosure policy,[4] and assures that the researcher gets its reward for submitting a valid report [24]. The platforms also validates and triages submissions using their own workers and contractors. The contractors are employed by the platforms' owning authority, some of whom are past researchers who transitioned from writing reports to validating reports [9].

There are three different use cases for the platform. One use case is so researchers can view the vulnerability disclosure policies, create and submit a bug report, and view all the public programs and any private programs from invitation. Another use case is so the vendors can update their vulnerability disclosure policy, view any submitted reports in their program, and update the status of reports. The final use case is for the platform itself: providing communication between vendor and researcher, updating the report once it has been validated and triaged, and sending funds to the researcher.

Program Management Process

The process in which vulnerabilities are found and reported is depicted in Figure 2.1, illustrating the interactions between vendors, platform, and researchers. The process is initiated by the vendor with a request to host a program on the platform. Before the

[4] In order for a bug-bounty program to be successful, a program must have rules and guidelines to be followed by the hacker and the vendor to ensure clear and concise communication and remediation of vulnerabilities. These rules and guidelines are written as the policy, and there is a standard policy established in the ISO/IEC 29147 standard

platform will publicize the program and send a notification to the pool of researchers, the platform requests the vendor send money to pay for future rewards and service fees.

The vendor sends a large sum of money to the platform and then the platform sends a promotion email to all the researchers, requesting them to participate in the vendor's program and find vulnerabilities. When a researcher finds a vulnerability, they create a report and submit it to the program. Then, the platform triages the report submission and determines if it is valid. If the report is valid, then the submission is updated and the vendor is notified that the report has been validated. It is then the responsibility of the vendor to review the report and determine a plan to fix/patch the vulnerability.

Once a fix has been determined, the vendor approves the submission and notifies the platform that the report is valid and tells the platform the reward amount for the found vulnerability. The platform takes their service fee from the pool and sends the reward to the researcher. Finally, the researcher receives their reward, and will continue looking for more vulnerabilities.

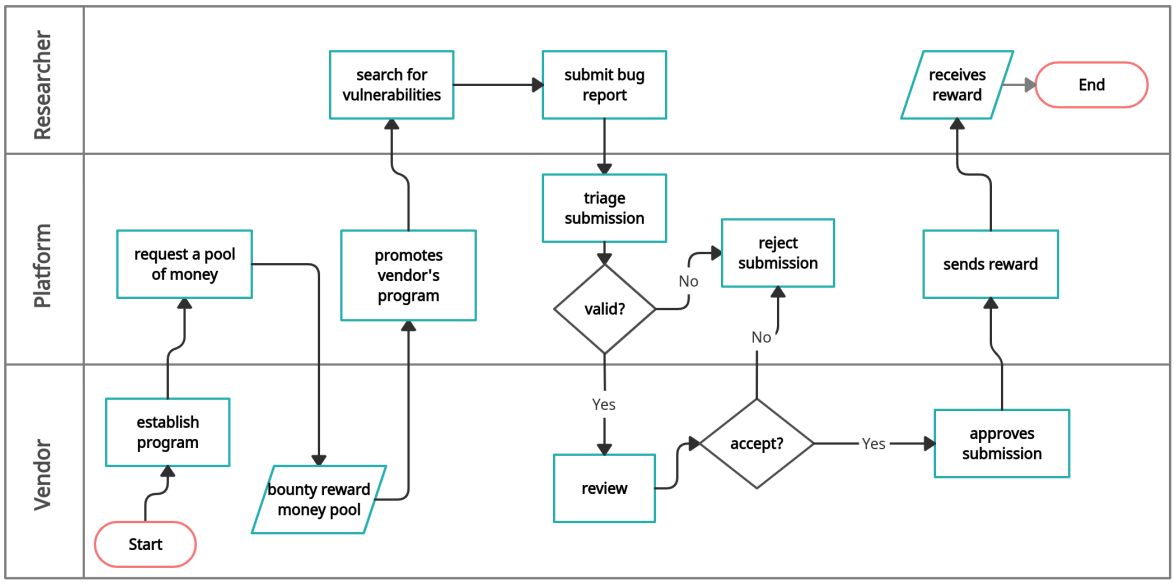

Figure 2.1: The process through which researchers, vendors, and platforms submit, manage, and maintain report submissions.

The platforms follow a set of standards that guide the platforms in managing bug-bounty programs and handling vulnerabilities submitted to the program.

Standards and Regulations

The difference between each platform is standards and regulations of compliance. For instance, HackerOne and BugCrowd comply to ISO/IEC 29147 and FedRAMP.[5] Because both platforms have government customers, they have to follow U.S. federal regulations.

In the European Union, there is an all encompassing set of regulations called GDPR[6] used for any company that has to store or transfer customer information. Below is an in depth explanation of relevant platforms.

Federal Risk and Authorization Management Program (FedRAMP)

FedRAMP is a standard required by any company that has a government organization as a customer and is only required in the United States. Companies from other countries that have U.S. government organizations as customers are also required to comply with FedRAMP. This standard ensures there is a plan in case of data breaches or other emergencies, and problems are solved in a timely manner. It also requires a continuous monitoring system to ensure that the necessary data is encrypted and not easily accessible to unauthorized actors.

National Institute of Standards and Technology (NIST) 800-53

FedRAMP uses NIST 800-53[7] as a baseline for their procedures and requirements. NIST 800-53 provides a catalog of security and privacy controls for information systems

[5] Federal Risk and Authorization Management Program
[6] General Data Protection Regulations
[7] National Institute of Standards and Technology standard 800-53 revision 5

and organizations to protect operations and assets.[8] These controls are flexible and can be adapted for any use.

General Data Protection Regulations (GDPR)

The E.U. GDPR focuses on the compliance of securing and transferring personal data and provides flexibility in how rules are used by companies. Certification of GDPR lasts for three years and can be withdrawn if requirements are not met during assessments [33].

ISO/IEC 29147: Information Technology - Security Techniques - Vulnerability Disclosure

This standard defines the reporting format used by every platform. However, not every platform complies with this standard. ISO-29147 is the guide on how to manage a program and format the reports. This standard pairs with ISO-30111 which explains how to receive vulnerability reports and process them.

The platforms use a tracking system to categorize report submissions as blocked, new, triage, resolved, or valid. While traditionally, the platform will triage the submissions for a vendor, a vendor's employees can still view the reports before the platform has reviewed the submissions. The reports are not encrypted on the platform, so all that is required to read and exploit vulnerabilities is a single sign-on.

ISO-27001

This standard, also known as the ISMS[9] standard, contains four phases of compliance: risk assessment, information security planning, security testing and evaluation, and certification and accreditation. This is the most popular ISO standard because compliance to this standard ensures the security of the organization's data and resources by implementing effective, time-verified security controls [15].

[8] https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/ nal
[9] Information Security Management System

Bug Reports

Bug reports (or vulnerability reports) are created by researchers to help vendors improve the security of their products and services. The report is considered 'valid' if it is within the scope of the program policy, follows the reporting format, and has not been reported before. Every platform adheres to a format for vulnerability reports as described in the ISO-29147 standards, with the major variation coming from the attachments in a report [16].

A bug-bounty report can contain:

- product or service name, URL, or affected version information;
- operating system of involved components;
- version information;
- technical description of actions being performed;
- sample code that was used to test or demonstrate the vulnerability;
- reporter's contact information;
- other parties involved;
- disclosure plan;
- threat/risk assessment details of identified threats and/or risks including a risk level for assessment results;
- software configuration of the computer or device configuration at the time of discovering the vulnerability;
- relevant information about connected components and devices if a vulnerability arises during interaction when a secondary component or device triggers the vulnerability;
- time and date of discovery; and
- browser information including type and version information [16].

The content of the report is significant because it can contain 'sample code' that demonstrates the vulnerability. However, this content option can be used to attach malware

to the report, opening up the vendors on the platform to an attack vector that is not referenced in any of the platforms, literature review, or standards and regulations complied with by the platforms.

Malware

Malware has more than one type of attack vector, it contains several options. **Malware**, or malicious software, is a file or script that is used to disrupt activity or cause mischief to the user [25]. There are nine different types of malware (there are more types of malware, but they are classified as subsets of the malware for this research.) It is important to provide the differences of each type of malware to illustrate the potential attack vectors.

Ransomware

Ransomware is malware that encrypts a victim's data or disables the functionality of the victim's computers until a payment is made to the attacker. A message may pop up on the screen that states "you have browsed illicit materials and must pay a fine," or the message may be in a foreign language or from a foreign police force [22]. If the payment is made, the victim receives a decryption key to restore access to their files. If the victim refuses to pay the ransom, then the hacker publishes the data on data leak sites (DLS) or blocks access to the files in perpetuity.[10]

Ransomware can be detected by monitoring for file entropy (file randomness). A file's entropy refers to a specific measure of randomness called "Shannon Entropy," where typical text files will have a lower entropy and encrypted or compressed files will have a higher entropy. In other words, by tracking files' data change rate, one can determine whether the file was encrypted or not.[11]

[10] https://www.crowdstrike.com/cybersecurity-101/malware/malware-vs-virus/
[11] Ransomware Detection Techniques

Spyware

Spyware is malware that attempts to silently monitor the behavior of users, record their web-surfing habits, or steal their sensitive data (i.e. passwords, personal identification numbers (PINs), and payment information) [2, 8]. The information is then sent to advertisers, data collection firms, or malicious third-parties for a profit.[12] Spyware tends to make a victim's computer sluggish and slow to respond. The best way to monitor for spyware is to test the computer's performance.

Adware

Adware is an automated, unwanted software designed to bombard users with advertisements, banners and pop-ups.[13] It infects a victim's computer using downloadable content deceptively enticing to victims. Adware makes advertisements appear in places where they normally would not. Monitoring for excessive appearances of advertisements is an effective way to detect adware.

Trojan

A Trojan is a digital attack that disguises itself as desirable code or software. Trojans may hide in games, applications, or even software patches. They may also be embedded attachments in phishing emails. By using some form of social engineering to trick the user into downloading the content, Trojans can take control of a victim's systems for malicious purposes such as deleting files, encrypting files, or sharing sensitive information with other parties [12].[14] A Trojan has several attack surfaces, and one way to detect the presence of a Trojan is by monitoring file activity. Files that are exported, modified, deleted, or altered when not authorized is a sign that a Trojan is present on the computer.

[12] https://www.crowdstrike.com/cybersecurity-101/what-is-spyware/
[13] https://www.crowdstrike.com/cybersecurity-101/adware/
[14] https://www.crowdstrike.com/cybersecurity-101/malware/trojans/

Worm

A computer worm is malware that attaches to a vulnerability host to replicate itself on that computer and continuously searches another vulnerability host which can be replicated onto another computer within the same network [11]. Typically, a worm spreads across a network through an Internet or LAN (Local Area Network) connection.[15] Computer worms are some of the oldest kinds of malware and can make a computer freeze or crash. Some ways to detect the presence of a worm is by monitoring for unusual computer or internet browser performance, and applications opening and running automatically.

Virus

A computer virus is a viral set that contains one program that reproduces itself [6]. Its definition is limited only to programs or code that self-replicates or copies itself in order to spread to other devices or areas of the network.[16] A virus will slow system performance, delay computer restart and shutdown, increase system crashes and pop-up of error messages.

Script-Based Malware

Malware is written as a script within a file that performs malicious actions. Script-based malware is most commonly used for web pages and documents in JavaScript [7].

Stego Malware

Stego malware, or stegware, is information hiding malware that uses Digital Stenography[17] to avoid detection [32]. Stego malware can be detected by monitoring for unauthorized uses of obfuscation. Obfuscation is the act of obscuring the file contents so they are not intelligible.

[15] https://www.malwarebytes.com/computer-worm

[16] https://www.crowdstrike.com/cybersecurity-101/malware/malware-vs-virus/

[17] Digital Stenography refers to the collection of techniques for hiding secret data inside innocuous looking digital media.

PUA/PUP

Potential Unwanted Applications, or Potential Unwanted Programs is a piece of software that is an option attached to free software downloads, or bundling software downloads. They may also appear as pop-ups and unwanted advertisements. PUA/PUP can be detected by monitoring the behavior of installed applications or programs for suspicious activity [19]. PUA and PUP are not classified as malware, but can be malicious just like malware.

MOTIVATION

This research focuses on characterizing attack vectors for bug-bounty programs because of the nature of bug-bounty platforms: a third-party company that offers a service for managing the vulnerabilities of other companies. These vulnerabilities are submitted in reports that consist of file attachments. Given the fact that the platforms and current set of standards do not mention 'malware detection', there is a necessity for improving the security of maintaining bug-bounty programs for vendors.

The motivation for looking into the security of bug reports is because they are under studied and there is no research looking into the relationship between file extensions and malware. This relationship is important with bug-bounty programs because of the potential for the disclosure of other vulnerabilities within the same program.

The reason for building a framework to characterize the attack vectors is because every outcome from the framework can be different for each vendor. Therefore, a conceptual framework is the best choice because of its flexibility in design and use.

Malware and bug bounty have been studied extensively, but the two have not been connected. I use a conceptual framework to connect files and malware by using the report's file attachments as malware files. The framework will characterize the potential attack vectors of malware allowed as a file attachment in a report. The best way to do just that is by using a conceptual framework. Chapter Six explains more about the conceptual framework.

Because the standard and regulations used by the platforms do not contain any suggestions for the safety of bug reports, I needed to find a security guide that would help to understand what security frameworks existed and what made them successful. How can I provide guidance to vendors without simply saying 'do not trust third party services' or 'make sure to use malware detection?'

Security Frameworks

The security gap in the platforms is due to the lack of malware detection for bug reports. There are a few cybersecurity frameworks that can address the security gap. The following frameworks are not used by platforms, but are a good starting point to address the gap in security. The three security frameworks to consider are NCSC, CIE, and NIST, all of which have overlapping security guidelines.

UK National Cybersecurity Centre (NCSC)

The NCSC[1] is a security framework established during 2008 in the U.K. for clients and customers that want the assurance that the data hosted by companies is secure. The NCSC is a popular framework used in like-minded countries, not just in the U.K [31].[2] The NCSC has also been established in the Netherlands and New Zealand [30, 35]. There are five secure design principles used in the NCSC framework; the following is a summary for each principle from the NCSC [21]:

1. Establishing the context: There is an understanding of the system operations and designs, use of third-party services and tools, and potential risks. There is a clause in this rule that specifically states that third-party services should not be fully trusted because an attacker can access a company's environment by gaining access through a third-party service environment.

2. Making compromise difficult: A secure system is characterized by applying methods and using techniques that make it harder for an attacker to compromise a company's system. The best way a company can protect itself from a security breach is by transformation[3], validation[4], and safe rendering.[5]

3. Making disruption difficult: Minimize access to certain interfaces to only those necessary. Privileged access is be done on a device that does not view messages or browse the internet. Monitor for security advisories and patches that need attention.

[1] National Cybersecurity Centre
[2] https://securityscorecard.com/blog/top-cybersecurity-frameworks-to-consider
[3] Transformation is taking one file format and transforming the file into a trusted format.
[4] Validation is checking to see if file and data structure are as expected for simple file formats
[5] Safe rendering means using a a virtual environment

4. Making compromise detection easier: Ensure that the system and services are available and functional.

5. Reducing the impact of compromise: Use a virus protection software and monitor for suspicious activity.

Cyber-Informed Engineering(CIE)

The CIE was developed by INL[6] to develop security solutions for an industrial application (more specifically, for nuclear and radioactive material facilities). There are eleven elements that make up the CIE framework [1]:

1. Consequence/Impact Analysis: Ways to impact availability or to steal data.

2. System Architecture: Data flow in the system.

3. Engineered Controls: Controls to mitigate cyber vulnerabilities.

4. Design Simplification: Reduce the complexity of digital design to the bare minimum that is absolutely necessary for critical functions.

5. Resilience Planning: Contingency planning and hardening of specific components or systems is necessary.

6. Engineering Information Control: Protect specific engineering records that are considered sensitive information (requirements, specifications, designs, configurations, analysis, and testing during system operation).

7. Procurement and Contracting: Outside vendors are held responsible for strong cybersecurity and collectively considered as part of the overall organizational cyber defensive posture.

8. Interdependencies: The system owner plans for risks introduced by the support of many disciplines, and understands the cybersecurity aspects of the interconnections.

9. Cybersecurity Culture: Cybersecurity is treated with the same rigor and attention as physical protection security.

10. Digital Asset Inventory: A complete inventory of the hardware, firmware, and software version levels of all engineering systems within the organization. Providing a mechanism for organizations to track and analyze these and the vulnerabilities residing in the software and hardware.

[6] Idaho National Laboratory

11. Active Defense: Resilient dynamic strategies and enhanced technical skill competencies to combat directed persistent attacks utilizing human behavior, supply chain, and state of the art technology.

National Institute for Standards and Technology (NIST) Cybersecurity Framework

This framework uses the word 'functions' instead of using the word 'principles.'[7] There are five functions in this framework:

1. Identify: An organization understands and manages the potential risks to data, systems, people, and assets.
2. Protect: Ensure data transfers and deliveries are secure.
3. Detect: Implement the necessary strategies and tools to monitor for suspicious activity.
4. Respond: Develop and implement remediation procedures to prevent cybersecurity events.
5. Recover: Back up any data in the event of a cybersecurity incident.

Within each function there are a set of categories that provide a more in depth coverage for each function. Each category has a unique identifier that provides a list of the characteristics of each function.

Tree for Cybersecurity Frameworks

The principles for NCSC, CIE, and NIST connect together as depicted in Figure 3.1. The framework for NIST and NCSC are activities to provide cybersecurity to a company, and the CIE framework acts as artifacts that connect with the activities.

[7] https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

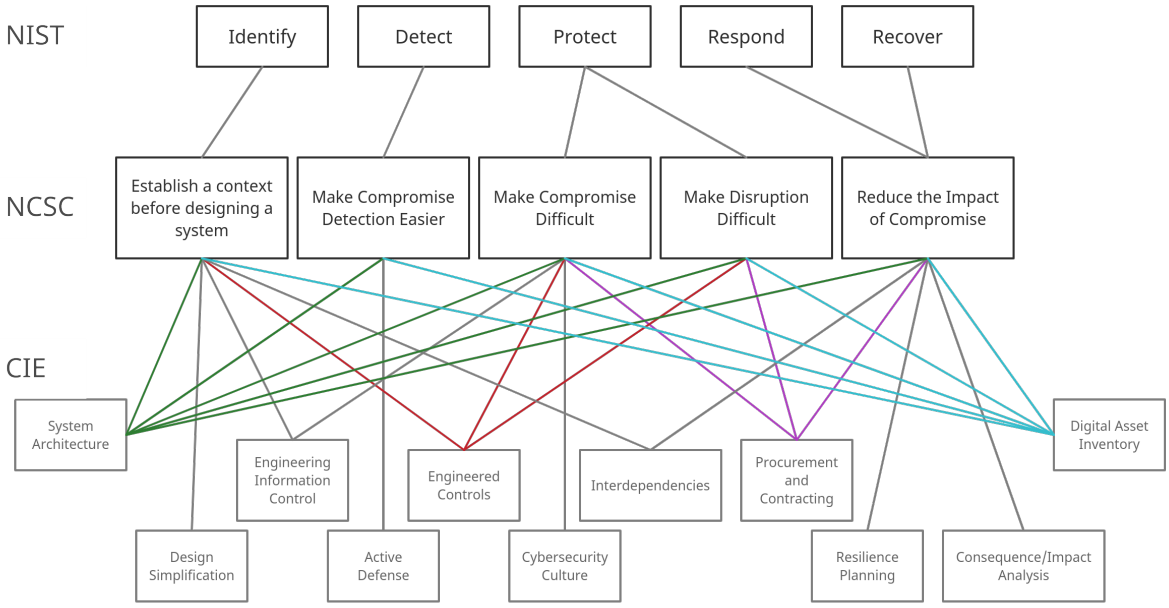

Figure 3.1: The Connections between NIST Cybersecurity Framework, NCSC Secure Design Principles, and the CIE Framework [10]

The NIST framework at the top carries down to the NCSC framework both of which represent activities to promote security. The final level, the CIE framework, is the artifacts that align with the activities that promote security. Four artifacts align with multiple activities and each activity consists of more than one security artifact.

Using the Connections

In each framework, there's at least one element that possesses the ability to provide protection to the host of a bug-bounty program. Within the NIST Cybersecurity Framework, we focus on the identity function; within the NCSC principle, we focus on 'establishing the context before designing a system'; and within the CIE framework, we focus on interdependencies, engineering information control, and engineered controls. The focus on these activities and artifacts align with the concept of malware detection for third-party service outputs. By understanding the potential attack vectors in the attachments of bug

reports, the vendors can create procedures for cautiously downloading and reading reports to avoid malware exposure.

Next, it was important to understand what tasks the platform provides for the vendor before being notified by the platform of a new report submission. Since these questions came from academia and not a customer, the questions were ignored. For that reason I requested assistance from vendors to get answers. From the help of both Medtronic and Workiva, I was able to perform an indirect interview and get an answer to these questions.

Interview Questions for Exploring the Security of the Platform

Post literature review, there were several questions whose answers were not immediately available on the platforms' websites. These questions are in the perspective of a company enquiry of hosting a bug-bounty program with the platform BugCrowd:

- Q1: Once a researcher submits a report, what's the procedure for sending the report to the vendor?
- Q2: Before notifying the vendor, who or what views the report? Is there use of a virtual environment to download or view the report?
- Q3: Is malware detection part of the report validation process?

BugCrowd's[8] response to the first question is similar to information that is found in two of their own articles [4, 5]: using diagrams to show the process of how reports get to the vendor. There is a team on the platform that ensures a report is valid (in scope of the program policy, reproducible, not a duplicate of a previous report, and poses a security risk). Once verified, they notify the vendor of the submission.

[8] A third-party company located in the United States.

This leads to the second question, before notifying the vendor, who or what views the report, and is there use of a virtual environment of some sort? The vendor's side of the platform can track the status of each report: new, blocked, valid, triaged, and done.[9] Therefore, this proves there is a potential to open a report with a status of 'new' before the platform has a chance to validate it. For this reason, the vendor must be required to follow a set of standards related to viewing the reports to stay secure; which there is no standard that contains requirements for malware detection (and there is no known reason why there is no malware detection in the standards). This response provides information for approaching the design of our conceptual framework.

From the platform's side of processing reports, the response from BugCrowd indicates that each submission is triaged and validated by a member of the triage team before it is sent to the vendor. This prevents the vendor from receiving non-valid (invalid, duplicate or non-applicable) reports. There is not an order for triaging and validating, but there is an indication that the ASE (Application Security Engineering) team uses a virtual environment to triage the reports. The use of a VM[10] during triage may prevent malware infections, but a platform's computer would not avoid malicious bug reports in the validation process.

Since there is no use of a virtual environment for report validation, is there any malware detection in the platform's procedure? BugCrowd avoided answering this question, creating the assumption that the platform does not scan reports for malware. This answer, or lack there of, leads to a followup question. Even if platforms did scan reports for malware, how does one scan for any malware when there are different types of file attachments in reports? The analyzing of malware samples is needed to understand the relationship between malware and file extensions, and to answer this question.

[9] `New': has not been processed by the platform. `Blocked': has an obstacle in the way that blocks a solution. `Triaged': platform has processed the report and a ticket has been created for the vendor. `Done': report has been fixed or patched.

[10] Virtual Machine

Because the third question was not answered, the assumption that malware detection is not done by the platform is acceptable. Therefore, an analysis of the relationship between malware and file extensions is necessary before an investigation into the allowed file attachments is done.

The data set of malware was gathered and sent by Hoplite Industries in three different ways. First the samples were gathered by downloading data from VirusTotal, a malware database that combines the file results from several malware detection companies. Hoplite also set up some honeypots to lure in malware samples. And third, the malware was collected from Hoplite's security connections. From this set of data, a script was written to categorize and move a smaller sample of malware data into a spreadsheet.

Analyzing Malware Samples

From looking at over 35 million different malware samples, a hypothesis was developed consisting of the types of malware that would be used as attachments in bug reports. Later, Chapter Five - Process - explains the correlation between the types of malware and file extensions that have been used in each type of malware.

Using a malware sample size of 15,734, I determined the relationship between malware and file extensions.

Table 3.1: Malware File Extensions Used in Study

Extension	File Description
(no extension)	Linux-Dev86 executable headerless
asm	assembler source ASCII text
asm	assembler source Non-ISO extended-ASCII text
asf	Microsoft ASF

Continues on next page

Extension	File Description
c	C source ASCII text
com	COM executable for DOS
com	COM executable for MS-DOS
dat	data
dex	Dalvik dex file version 035
dll	PE32 executable (DLL) (console) Intel 80386
dll	PE32 executable (DLL) (GUI) Intel 80386 for MS Windows
dll	PE32 executable (DLL) (native) Intel 80386 for MS Windows
dll	PE32+ executable (DLL) (GUI) x86-64 for MS Windows
doc	Composite Document File V2 Document Can't read SSAT
doc	Composite Document File V2 Document Little Endian Os
dos	DOS batch file ASCII text with CRLF line terminators
dos	DOS batch file ISO-8859 text
dos	DOS executable (COM)
elf	ELF 32-bit LSB executable Intel 80386 version 1 (SYSV)
exe	PE32 executable (console) Intel 80386 for MS Windows
exe	PE32 executable (GUI) Intel 80386 for MS Windows
exe	PE32 executable (native) Intel 80386 for MS Windows
exe	PE32 executable Intel 80386 for MS Windows
exe	PE32+ executable (GUI) x86-64 for MS Windows
gif	GIF image data version 89a 10 x 12
gz	gzip compressed data from NTFS filesystem (NT)
gz	gzip compressed data from Unix

Continues on next page

Extension	File Description
html	HTML document ASCII text
html	HTML document ISO-8859 text
html	HTML document Non-ISO extended-ASCII text
html	HTML document UTF-8 Unicode (with BOM)
img	DOS/MBR boot sector code offset 0x3c+2 OEM-ID "MSDOS5.0"
iso	ISO-8859 text with CRLF line terminators
iso	Non-ISO extended-ASCII text
jar	Java archive data (JAR)
java	compiled Java class data
jpg	JPEG image data JFIF standard 1.01 aspect ratio density 1x1
ms	MS Windows shortcut Item
pas	Pascal source ASCII text
pdf	PDF document version 1.3
php	PHP script ASCII text with CR line terminators
pl	perl script ASCII text executable
pl	perl script executable (binary data)
png	PNG image data 8-bit/color RGBA non-interlaced
rar	RAR archive data
rtf	Rich Text Format
smgl	exported SGML document ASCII text
smtp	SMTP mail ASCII text with CRLF line terminators
swf	Macromedia Flash data (compressed)
txt	ASCII text

Continues on next page

Extension	File Description
txt	UTF-8 Unicode text
xls	Composite Document File V2 Document Little Endian Os 0
xml	XML document text
zip	Zip archive data

The types of malware chosen are based on the findings in Table 3.1 - Adware, Ransomware, Spyware, PUA/PUP, script-based malware, Trojan, Virus, Worm, and Stego malware. Though Chapter Two defines the selected types of malware, this table supports the reason why this set of malware was chosen. All will make sense in Chapter Six when depicting the relationship between file extensions and malware, but at this point, know that the malware was chosen based on the behavior of each malware type and the file extensions used for that type.

The other reason this set of malware was chosen is because their initial attack vectors are unique from each other, however malware of one type can be used to download and spread malware of another type. Also, some malware types are subsets of another type, for instance, a Trojan has several subsets like clickjack, exploits, and packers as will be shown in Chapter Seven. Finally although PUA/PUP is not technically defined as a malware, there is malware the is represented as PUA/PUP, like grayware.

GQM (GOAL, QUESTION, METRIC)

Following Soligen's 2002 GQM [34][1], the aim is to achieve the goal by answering a set of questions following a specific set of metrics. These metrics help answer questions, which in turn helps accomplish the goal of this research. The specific goals, questions, and metrics of this research are presented below:

Goal 1

Investigate bug-bounty programs *for the purpose of* providing a conceptual framework *with respect to* malware detection in vulnerability reports *from the perspective of* reviewing reports *in the context of* verifying no malicious attachments exist in the reports.

Goal 2

Investigate malware *for the purpose of* finding its relationship with file extensions *with respect to* the file attachments allowed by a bug-bounty platform *from the perspective of* a vendor(current or prospective) *in the context of* characterizing potential attack vectors.

Questions

Q1: Are there any standards and regulations implemented in the platforms that prevent malware infection through a vulnerability report?

Q2: Using the data collected, what file types are most commonly used for the platforms and what is the type of malware that corresponds with that file type?

Q3: By illustrating the relationship between file extensions and malware types, can the conceptual framework improve the ability for vendors to assess the safety of vulnerability reports?

[1] Goals, Questions, Metrics

Answering these questions determines if there is a way to improve upon the current security process in bug-bounty programs.

Metrics

M1: Malware Type (Q1, Q2, and Q3)

M2: Standards and Regulations (Q1)

A: ISO Standards, **B**: FedRAMP, **C**: GDPR

M3: File Type (Q2)

M4: Number of Platforms (Q1)

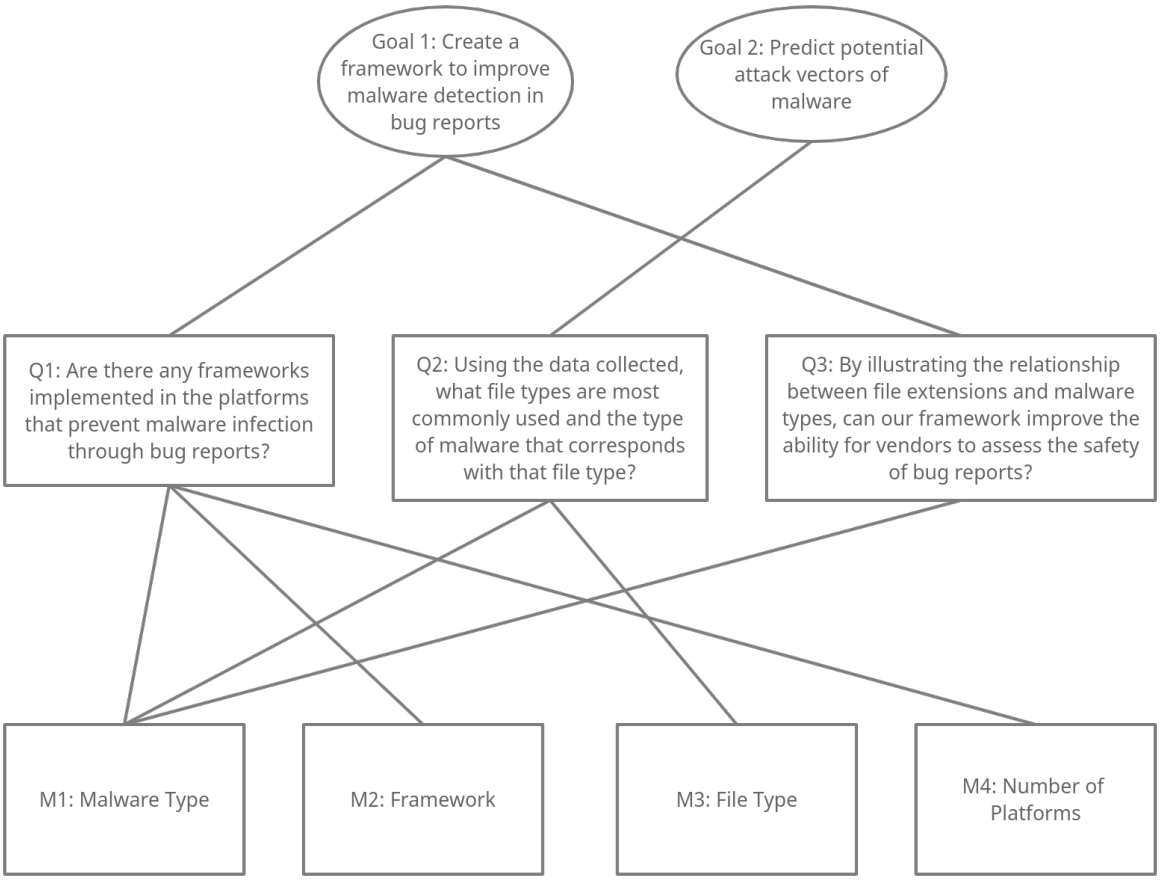

Figure 4.1: Research Goals Questions Metrics for Building the Conceptual Framework

PROCESS

This chapter provides the procedure used to collect all the platform data and portray that data in tables. The research process uses inductive, abductive, and deductive approaches. Abduction is defined as making a conclusion from what is known;, induction is making an inference from an observation of the data.[1] The abductive portion of this research is done using the literature review and the interview from a bug-bounty vendor, helping to establish the knowledge necessary for the collection of data. The inductive portion of this research is done by surveying each bug-bounty platform, which helps form the GQM (Goals Questions Metrics). The GQM is used as an approach to analyze and refine the theory of this conceptual framework using the deductive approach. A conceptual framework is assembled using the results from this data collection and analysis.

The following information was gathered from each platform: services, geographical location, standard compliance, registration information on researchers, year established, file size and count, and file type restrictions. After gathering this data, it was reduced and categorized for the study. The goal of the data collection was to gather enough information to build the conceptual framework and understand the relationship between malware and file extensions.

Heuristics for Data Collection

With the background knowledge acquired, careful and thoughtful research questions were created that assist in identifying components in this observational study [36]. These questions follow an inductive approach that helps determines the data to collect in the investigation.

[1] Merriam Webster Dictionary De nition

Preliminary Guiding Questions

- PQ1: Is there a malware detection procedure for bug-bounty programs?
- PQ2: Are there differences between platforms located in the U.S. and other countries?
- PQ3: Are there types of files that malware can be attached to in a report?

Data Collection Process

The investigation began by exploring each platform and collecting data from its own bug-bounty program as the sample. To aid the selection of platforms, a repository on GitHub was used to provide a list of different bug-bounty platforms used around the world.[2] Also, a Google search of different bug-bounty platforms was performed to create a final list of all the platforms. Table 5.1 identifies the geographical location of the platform, their advertised compliance to specific standards and regulations, the required information to register as a researcher, and the service options available. The following list of abbreviations is used in the table:

BB: Bug-Bounty Program

VD: Vulnerability Disclosure Program

PT: Penetration Testing

RT: Red Teaming

DoB: Date of Birth

Basics: First and Last Name, Username, and Email

Table 5.1: Third-Party Platform Data Collection on Registration Information, Geographical Location, Standard Compliance, and Service Options for all Identified Platforms

[2] https://github.com/disclose/bug-bounty-platforms

Platforms	Location	Stnds/Regs	Personal Info	Services
Antihack.me	Singapore	N/A	N/A	BB, PT, RT
BB Caribbean	Caribbean	ISO-27001	Application	BB
BB Switzerland	Switzerland	GDPR	Application	BB
BugHunt	Brazil	None	Basics	BB, VD
BugBase	India	ISO-27001	Country of Origin	BB, PT, VD
BugBounter	Turkey	GDPR	Basics	BB
bugbounty.jp	Japan	ISO-29147	Basics	BB
bugbounty.sa	Saudi Arabia	N/A	Bank Account, Tele #	BB
bugbounty.ru	Russia	ISO-27001	Basics	BB
BugCrowd	US	FedRAMP	Basics	BB, VD
Bugv	Nepal	None	Basics	BB, PT, VD
CrowdSwarm	UAE	N/A	DoB, Tele #, Addr, Passport	BB, PT
Cyber Army Indonesia	Indonesia	N/A	Indonesia Citizenship	BB, VD
Cyscope	Chile	None	Country	BB
Detectify	Sweden	GDPR	TBD	VD
EpicBounties	Spain/LATAM	GDPR	DoB, Addr Passport	BB
Federacy	US	SOC2	Basics	BB, PT, VD
FindBug	Kosovo	GDPR	Complete CTF	BB, VD

Continues on next page

Platforms	Location	Stnds/Regs	Personal Info	Services
Frontal	UAE	N/A	Application	BB, PT, VD
GObugfree	Switzerland	GDPR	Basics	BB, PT
HackenProof	Estonia	GDPR	Basics	BB, PT
HackerOne	US	FedRAMP	Basics	BB, PT, VD
Hackrate	Hungary	GDPR	Basics	BB, PT, VD
HACKTIFY	Hungary	GDPR	Basics	BB, PT, VD
Hackrfi	Finland	GDPR	Basics	BB
Immunefi	N/A	None	Basics	BB
Inspectiv	US	FedRAMP	Basics	VD
Intigriti	Belgium	GDPR	Basics	BB
Open BB	Bangladesh	ISO-29147	Twitter Account	BB
Ravro	Iran	ISO-29147	Basics	BB
RedStorm	Singapore	None	Basics	BB, VD
SafeHats	India	ISO-27001	Basics	BB, VD
Safevuln	Vietnam	None	Basics	BB
Secuna	Phillipines	ISO-29147	DoB, Addr, PoI Nationality	BB, PT, Audit
SecureBug	Sweden	GDPR	Basics	BB, PT, RT
SlowMist	China	N/A	Application	BB, BC, RT
Swarmnetics	Singapore	N/A	Application	BB, PT, VD
Synack	USA	ISO-27001	Application	BB, PT, VD
TestBirds	Netherlands	GDPR	Loc, DoB	BB, PT
TheBugBounty	Malaysia	N/A	N/A	BB, PT

Continues on next page

Platforms	Location	Stnds/Regs	Personal Info	Services
v1bounty	Germany	GDPR	Basics	BB, VD
Vulbox	China	None	Tele#(+86)	BB, VD
Vulnerability Lab	Germany	GDPR	Basics	BB
Vulnscope	Chile	N/A	Country, DoB	BB, PT
WhiteHub	Vietnam	None	Basics	BB, PT
YesWeHack	France	GDPR	N/A	BB, VD
Yogosha	France	GDPR	Community	BB
Zerocopter	Netherlands	GDPR	N/A	BB, VD

Registering for Different Platforms

Using the data collected from the different platforms, the list was reduced to platforms with a minimum requirement for registration information (first and last name, an email address, and a username). This reduction was done due to the time and effort it would take to create an account and submit a bug report in a malicious scenario.

Once registered on the platform, the programs hosted on each platform were investigated. Some of the surprising findings were the same vendors being hosted on different platforms[3] and American companies were hosting programs on platforms from other countries. For example, Sony is hosting a public program on an Asian platform.

This is the final list of platforms used in this study:

[3] Medtronic was one of these companies that had programs on HackerOne and BugCrowd. This was discovered in looking at Medtronic's vulnerability disclosure website and communicating with Medtronic's security team about their bug-bounty program

Table 5.2: Selected Third-Party Platforms with Geographical, Standard Compliance, and Year Established

Company	Location	Year Est.	Compliance
BugHunt	Brazil	2020	None
BugBase	India	2021	29147
BugBounter	Turkey	2020	GDPR
BugBounty.jp	Japan	2015	ISO-29147
BugBounty.ru	Russia	2021	ISO-27001
BugCrowd	US	2012	FedRAMP
bugv	Nepal	2020	None
Federacy	USA	2018	SOC2
HackenProof	Estonia	2017	GDPR
HackerOne	US	2012	FedRAMP
HackRate	Hungary	2020	GDPR
Hackerfi	Finland	2016	GDPR
HACKTIFY	Hungary	2020	GDPR
Immunefi	Virtual	2020	GDPR
Inspectiv	US	2018	FedRAMP
Intigriti	Belgium	2016	GDPR
Ravro	Iran	2019	N/A
RedStorm	Singapore	2018	None
SafeHats	India	2012	ISO-27001
SafeVuln	Vietnam	2014	None
SecureBug	Sweden	2019	GDPR
WhiteHub	Vietnam	2019	None

File Attachments with Vulnerability Reports

I registered for 22 platforms to identify the restrictions and file types for report attachments. By clicking a link or button that says 'submit report' or 'create a report,' every one of the platforms has a similar reporting format in how the researcher is to complete a valid bug report.

Each platform requested the same elements within the report [16]:

- A description of what service or product is affected by the vulnerability.
- How the vulnerability can be identified, demonstrated, or reproduced.
- The impact the vulnerability has on a service or product.

The difference between the platforms lies in the size, count, and type of file attachments. When a file type is not allowed, a researcher can not upload that file to the report. When the size of a file is too big, or the combination of files size is over the limit, a notice appears above the report with a message such as 'file size over limit.'

The following is a collection of the file size and count restrictions in each platform's report:

Table 5.3: Third-Party Platforms Report Attachment Restrictions on Size and Count

Company	Location	Max Size	Max Count
Bug Hunt	Brazil	No Limit	10
Bug Base	India	25	No Limit
BugBounter	Turkey	50	1
bugbounty.jp	Japan	5	5
BugBounty.ru	Russia	30	No Limit
BugCrowd	US	50	20
Bugv	Nepal	5	5
Federacy	USA	N/A	N/A
HackenProof	Estonia	50	5
HackerOne	US	250	No Limit
HackRate	Hungary	150	No Limit
HACKTIFY	Hungary	250	No Limit
Hackerfi	Finland	No Limit	1
Immunefi	Virtual	8	20
Inspectiv	US	10	20
Intigriti	Belgium	5	30
Ravro	Iran	100	No Limit
RedStorm	Singapore	No Limit	No Limit
SafeHats	India	No Limit	No Limit
Safevuln	Vietnam	50	3
SecureBug	Sweden	50	10
WhiteHub	Vietnam	50	10

Next, I implemented tests to identify the types of files that are allowed as report attachments. File extensions were chosen based on the findings from the set of metadata of malware samples (i.e. over 35 million). All the file extensions were categorized into a general set of file types. The file types were split into these categories: images, videos, links, executable, compressed, documents, and PDF.

By examining the metadata of the malware machine, 95 unique file extensions were identified, and of those, 40 contributed to at least 1000 samples of the same file extension. After gathering data on each of the file extensions that were considered 'significant', the file extensions were organized into categories based on similarity. For example, since *jpg* and *png* are file extensions that use an image format, they are grouped together in an 'image' category. These extensions are tested on the selected platforms. Any category that contains at least one type of file extension is classified as an attachment for each platform.

These extensions were categorized and tested:

Image: bmp, dex, fpx, gif, jpg, png, riff, swf

Videos: m2t, mp4, mts

PDF: pdf

Document: doc, docx, ppt, xls, xml, rtf

Executable: (no extension), dll, exe, php, pl, py, sh, rb

Compressed: gz, rar, tar, zip

Links: chm, html, lnk

Of all the platforms examined, only one did not have any options for file attachments: Federacy, as depicted in Table 5.4. This means a hacker is not able to attach malware to a vulnerability report. Thus, showing that every platform is susceptible to malware infection when security precautions are not in place.

After testing, this was the results on each platforms accepted file types:

Table 5.4: Accepted File Types on Bug-Bounty Platforms

Company	Img	Vid	PDF	Doc	Exe	Zip	Link
Bug Hunt	p	p	p	p	p	p	
Bug Base	p	p	p			p	
BugBounter	p	p	p				
bugbounty.jp	p						
BugBounty.ru	p	p	p	p	p	p	p
BugCrowd	p	p	p	p	p	p	p
Bugv	p						
Federacy							
HackenProof	p	p	p	p			
HackerOne	p	p	p	p	p	p	p
HackRate	p	p		p			
HACKTIFY	p	p	p	p	p	p	p
Hackerfi	p		p				
Immunefi	p						p
Inspectiv	p	p	p	p	p	p	p
Intigriti	p	p	p	p			p
Ravro	p	p	p	p		p	
RedStorm	p		p	p			
SafeHats	p	p	p	p			
Safevuln	p	p	p	p			
SecureBug	p		p				
WhiteHub	p	p	p	p	p	p	p

Current Malware Detection Plans

Based on PQ1 (preliminary-guiding question), the overall answer is that there are no scans or detections performed on to a bug report before opening and validating the report. This is because bug-bounty platforms use a web-based report-tracking system to organize all submissions in a bug-bounty program. Other than using a basic virus scanning tool, there is no plan in place for scanning each report for malicious scripts or malware.

Differences in Platforms

In PQ2, there are several differences between platforms from the U.S. and other countries. For example, any platforms originating from the Americas or Asia do not use GDPR[4], because this is a regulation of the European Union. Each platform also offers a different set of services to vendors, not just bug-bounty programs. In addition, each platform collects a different set of information about the researcher. Some need a username, full name, and email (which could be easily falsified). Other programs require additional, personal information such as country of origin, bank account information, passport number, address, and phone number. Some programs even restrict who is allowed to join a platform, meaning they only allow people within the country of origin to join their platform. Appendix A contains tables with the results from this data collection.

These results show that most platforms follow some sort of standard or regulation. The main ISO standards are ISO-27001 and ISO-29147, and the main regulations are GDPR and FedRAMP. If a platform does not comply with a standard or regulation, there is usually more identification required to join the platform. However, there are still a handful of platforms that do not comply to any standards or regulations, and only require the basics to sign up as

[4]General Data Protection Regulations

a researcher. This makes these platforms a prime target for hackers to exploit the platform.

Relation of Malware and File Types

I wanted to analyze each platform and figure out the types of malware to which the vendor may be exposed by the report attachments. It is crucial to know what types of files can have malware attachments in order to assess the potential attack vectors. A more in-depth analysis of this is done in a case study in the next chapter.

Of all the platforms in this study, all but one allows for images. Images consist of file extensions such as *gif, jpg, png* to name a few. The next most commonly accepted file type is PDF; 17 platforms accepted *pdf* as file attachments. Videos are accepted as a report attachment in fifteen platforms. Video files include *m2t, mts, mp4* file extensions. Only fourteen of the platforms allow for text-based documents. Text files include *ppt, doc, xml, rtf* file extensions.

Only nine of the platforms allow compressed files in a bug report. A compressed file has several different extensions, but was tested with a *zip, rar*, and *gz*. A compressed file is the second most common way to embed malware[5]; this is due to the lack of visibility of the contents of a zipped file prior to being downloaded. A zipped file does not need to be extracted before malware can infect a computer, the file can spread malware in its zipped form.

Links are allowed as attachments in eight platforms. The links were tested using *html, lnk* and *chm* file extensions. Only seven platforms allow for scripts, which is the most common way to spread any type of malware. The script extensions that were tested include *dll, exe*, and scripts from programming languages like Python, Java, Perl, and Ruby. If a platform allows for scripts, the reports would have to be scanned for malicious scripts before

[5] evidence from VirusTotal

downloading the report. If not, for these seven companies, this is a major risk for potential exploitation.

ANALYSIS

This chapter provides an analysis of the data, answers to the GQM questions, and illustrates the findings from the observational case study. The analysis of the data provides the answers to the GQM questions. Then a study of malicious files embedded into bug reports is applied to create a conceptual framework for characterizing attack vectors. The case study validates the decisions for file extensions and types of malware used in the study.

Observational Case Study

According to Yin's Case Study Research, a case study is used with the goal of testing the different ideas that would contribute to the final solution [36]. This case study is used to formulate the framework and answer the questions from the GQM. This study answers the 'what', 'why', and 'how' a conceptual framework is a solid solution for characterizing potential attack vector in bug reports.

So far, the study has provided information about the chosen bug-bounty platforms, divided the file extensions into seven categories, identified the three cybersecurity frameworks that could be used to improve the security for bug-bounty programs, identified the types of malware as potential report attachments, and identified the file extensions that are allowed in reports. Following is an illustration of the relationship between file extensions and malware using a large sample size of malware.

The malware samples were provided by Hoplite Industries. Using the malware sample metadata, I was able to calculate the relationship between file extensions and malware.

All the metadata was sourced from a large database and information was retrieved by using SQL[1] queries. These queries produced a list of all the file extensions used in over 35 million malware samples. For this case study, the list was reduced to file extensions that

[1] Sequence Query Language

had more than ten samples in a file extension.

Table 6.1: Data Collection of File Extension and Count on Malware Samples with More than 20 Samples per Extension

Extension	Count	Extension	Count	Extension	Count
exe	18318271	html	10175562	(no ext)	2813930
dll	2180563	zip	690314	pdf	291078
rar	246302	m2t	183796	gz	82844
jpg	77256	doc	71797	dex	54009
lnk	29421	png	28290	xml	20309
gif	18915	php	18192	xls	17697
fpx	16102	xlsb	12898	swf	11072
sh	4972	pl	4797	rtf	4311
mp3	3619	bmp	3407	chm	2366
docm	2354	xlsm	1827	mp4	1691
riff	1310	mts	1072	asf	836
bz2	762	plist	662	py	645
wmf	608	docx	595	tar	582
ttf	512	ogg	462	xlsx	417
ppt	415	wav	356	tif	339
svg	229	(386)	207	a	207
torrent	145	mov	110	rsrc	101
eps	92	dotm	76	m4a	59
rb	55	xlam	53	webp	49
otf	43	mpg	31	pptx	26
webm	23	acr	21		

With the file extension collection, the list was categorized and reduced to file extensions with no less than 1000 samples or commonly used file extensions:

Table 6.2: Selected File Extensions with File Count and Category(Type)

Extension	Count	Category	Extension	Count	Category
exe	18318271	executable	html	10175562	link
(no extension)	2813930	executable	dll	2180563	executable
zip	690314	compressed	pdf	291078	Adobe
rar	246302	compressed	m2t	183796	video
gz	82844	compressed	jpg	77256	image
doc	71797	document	dex	54009	image
lnk	29421	link	png	28290	image
xml	20309	document	gif	18915	image
php	18192	script	xls	17697	document
fpx	16102	image	xlsb	12898	document
swf	11072	image	sh	4972	script
pl	4797	script	rtf	4311	document
mp3	3619	audio	bmp	3407	image
chm	2366	link	docm	2354	document
xlsm	1827	document	mp4	1691	video
riff	1310	image	mts	1072	video
py	645	script	docx	595	document
tar	582	compressed	ppt	415	document
mov	110	video	rb	55	script

Once there was an adequate sample size of each file extension, the type of malware

was identified for that file extension. This data is used to create a diagram that shows the relationships between file types and malware types.

Table 6.3: Relationship between Types of Malware and File Extensions

File Ext	Ad	PUA/P	Spy	Script	Trojan	Ransom	Virus	Worm
no ext							*p*	
bmp					*p*			
chm						*p*		
dex					*p*			
dll	*p*				*p*		*p*	*p*
doc					*p*			
docx						*p*		
exe	*p*	*p*	*p*		*p*			
fpx					*p*			
gif					*p*			
gz	*p*		*p*		*p*			
html			*p*	*p*	*p*			*p*
jpg					*p*			
lnk					*p*			
m2t						*p*		
mp3			*p*		*p*			
pdf					*p*			
php				*p*		*p*		
pl				*p*	*p*			
png			*p*		*p*			

Continues on next page

File Ext	Ad	PUA/P	Spy	Script	Trojan	Ransom	Virus	Worm
ppt			*p*		*p*			
py				*p*				
rar	*p*	*p*			*p*			*p*
rb				*p*				
riff								
rtf					*p*			
sh			*p*	*p*	*p*			*p*
swf					*p*			
tar								
xls								
xml					*p*			
zip	*p*				*p*	*p*	*p*	*p*

The goal of this observational case study was to create a visual of the relationship between types of malware and file extensions. This was done by using malware samples as evidence to back up how this conceptual framework characterizes the attack vectors for each platform. This case study validates the legitimacy of the behavioral diagram in this conceptual framework and provides proof to answer questions from the GQM.

Q1: Are there any standards and regulations implemented in the platforms that prevent malware infection through a vulnerability report?

There are no standards and regulations in place on the platforms that contain a clause or requirement for malware detection. This oversight opens up the potential for non-public bug reports to be exposed to the public, sold on the black market, and used to exploit a

company. This answer is supported by reviewing every standard and regulation used by the platform to ensure the phrase 'trust a third party services' or 'malware detection' does not exist in the current regulations.

Q2: Using the data collected, what file types are most commonly used and what is the type of malware that corresponds with that type?

By analyzing the collection of data, the connection between malware and file types is illustrated in the next section. However, one important detail to note is that use of executable and script files makes a company susceptible to every type of malware in our case study (i.e., Adware, Spyware, Ransomware, Virus, Worm, Trojan, PUA/PUP, and Stego malware). The answer to the second question in the GQM is that images are the most commonly allowed file type between the platforms. This leaves Trojans and Stego malware as the only potential types of malware used in reports based on our study. This answer is supported from the findings in table 5.4,[2] 6.2,[3] and 6.3.[4]

Q3: By illustrating the relationship between file extensions and malware types, can this conceptual framework improve the ability for vendors to assess the safety of bug reports?

The framework is used to improve the ability for vendors to choose the best platform that balances report attachment options with security by characterizing the potential attack vectors. This is done by providing the ability to characterize the potential attack vectors for each platform. The framework provides an illustration of the structural and behavioral guide of the bug-bounty report verification process. The structural diagrams illustrate the connections between the components and the behavioral diagram guides the vendor with a

[2] Accepted File Types for Bug-Bounty Platforms
[3] Malware Sample File Extensions, Count, and Category
[4] Relationship between Malware Types and File Extensions

step-by-step guide on how to use the framework, and how each component interacts.

CONCEPTUAL FRAMEWORK

The objective for this conceptual framework is to help vendors choose a platform by assessing the potential attack vectors based on the bug-bounty platform. This is to be used as a tool to assist vendors. This is not to be used to detect malware, but to understand the potential attack vectors that are introduced by different types of malware.

A conceptual framework is an overarching argument about the importance of the research and how different concepts or theories are connected. It defines not only why an investigation is important but also how it should be done [23]. The framework is portrayed as figures and diagrams, illustrating the connection between concepts and the importance of the study. There is no standard format for the design of a framework, only the overall purpose of the framework.

The purpose of a conceptual framework is to articulate the logical connection between the problem identified and the methodological innovations, all of which can evolve over time as the understanding of the material increases [23]. This framework is used to show the connections between concepts and create thoughtful questions that help reach an answer to the overall problem that exists in a study. Since this study relates to malware detection, the purpose of this framework is to create a procedure for safely handling bug reports by assessing potential attack vectors from malicious report attachments.

A conceptual framework was chosen because of its use in research to combine well-studied components to create a new theory: to show that there has been plenty of research done for each of the necessary components, but these components have not been connected before. In this framework, file attachments, malware, and bug-bounty platforms are joined together.

There is a plethora of studies on malware available through academic articles and security companies. VirusTotal is used to gather data about the file extensions that have

been used for malware, which is the source of the list of malicious file identities. The use of existing malware articles and websites that have trustworthy malware detection software provides validity to the decision to use the information found in these resources.

Investigating the cybersecurity frameworks from Chapter Three and the standards and regulations used by the platforms is evidence that malware detection is not a concept that has been considered in this context. Instead, the standards recommend the platforms have a plan of action in place in the event of a data leak. This investigation into the procedures and activities of the platforms shows that malware detection or scanning file attachments is not specified in any platform.

Therefore, this conceptual framework provides an illustration of the relationship between malware, file attachments, and the potential attack vector used to exploit bug-bounty platforms.

Components of the Conceptual Framework

In a conceptual framework the components are used to make up a theory or phenomenon that creates the final product (results of the study). The components in the conceptual framework are restated after the exploration and analysis of the data: file attachments, malware types, and bug-bounty platform. The following explains the importance and role of these items:

File Attachments

During the research process, files were sorted into seven categories: Documents, Images, Videos, Compressed, Executable, Links, and PDF. This was done to create a clean visual that can be easily explained. During the case study, the relationship between file types and malware types was analyzed.

Image: bmp, dex, fpx, gif, jpg, png, riff, swf

Videos: m2t, mp4, mts

PDF: pdf

Document: doc, docx, ppt, xls, xml, rtf

Executable: (no extension), dll, exe, php, pl, py, sh, rb

Compressed: gz, rar, tar, zip

Links: chm, html, lnk

Malware Types

The purpose and results of the case study validate the decision to include malware types as a component of this framework. There are many different types of malware, but for illustration purposes, it is narrowed to the common types of malware that were found in over 35 million samples. These nine types of malware cover all types of files that can be attached to a report and have a different attack vector. The different types of malware that are used in this case study are Adware, PUA/PUP, Ransomware, Spyware, Trojan, Virus, Worm, Stego and Script-Based malware.

Bug-Bounty Platforms

The platform is included as a component of the framework because there is no correlation between the standards followed and the types of files accepted by the platform. This correlation analysis is done by comparing tables 5.2 and 5.4, and combining the data to illustrate how each standard relates to the file types. It's important to note that not every platform that complies to FedRAMP or GDPR allows for every file type, and that there is an overlap. For the platforms that follow each compliance, the following table indicates the potential file type allowed.

Table 7.1: The Correlation Between File Types and Compliance

File Type	ISO-29147	ISO-27001	FedRAMP	GDPR
Images	*p*	*p*	*p*	*p*
Videos	*p*	*p*	*p*	*p*
Documents		*p*	*p*	*p*
PDF	*p*	*p*	*p*	*p*
Compressed	*p*		*p*	*p*
Executable			*p*	*p*
Links			*p*	*p*

Discussion

The takeaways from the case study is an understanding of the files types used by different types of malware. In addition, there is a suggestion on the location to modify the current ISO-29147 standards and hypothesize on how the framework works when presented to a vendor.

ISO Modification Recommendation

If there is a place in the current report handling process that could improve the security for vendors when handling reports, it is in the verification step. By assuring the report is legitimate (not only in the content of the report, but also in the attachments) the report soundness is ensured.

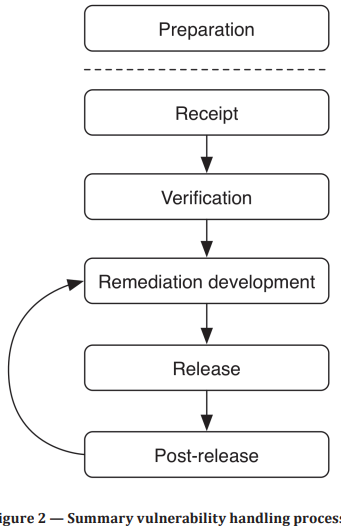

Figure 7.1: Summary of Vulnerability Handling Process from ISO-29147[16]

Hypothesis of Conceptual Framework

Trying to cover the entire attack surface for all malware is impractical, so the goal is to make the attack surface manageable. However, by knowing what files extensions are used, the attack surface is reduced to a manageable size. The goal for this conceptual framework is to lead a vendor through the process of understanding the types of malware used in their platform's bug reports to assess the potential attack vectors (based on the current knowledge obtained from researching these bug-bounty platforms). By understanding the potential attack vectors, the vendor can monitor computer functionality and monitor for the modification of files or other data.

From the GQM, the goal is to add the procedure of malware detection specifically in vulnerability reports and characterize potential attack vectors. Therefore, the contribution from this research is understanding that there are trade-offs to having restrictions on the report's file attachments - convenience or security. This conceptual framework is meant to be used by a vendor whose goals is to improve the security of bug reports. The conceptual framework is compiled to link all the different components necessary to reach this goal.

Justifying The Theory

The theory of this conceptual framework is the following: an understanding of file types can help identify types of malware and thereby characterize the potential attack vectors introduced by a vulnerability report.

In order to assemble a conceptual framework, it is necessary that the evidence is organized and structured into a concise and precise manner [29]. This is done by performing studies to validate the theory(s).

According to Shull and Sjoberg [27, 29], the process to building a theory involves constructs, propositions, explanations, and scopes. The theory is built using the following steps:

1. Defining the Constructs of the Theory
2. Defining the Propositions of the Theory
3. Providing Explanations to Justify the Theory
4. Determining the Scope of the Theory
5. Testing the Theory Through Empirical Research

Building the Theory

In the context of this idea, these are the core assumptions followed:

The vendor does not use a virtual environment to open reports.

The vendor believes the researcher has no malicious intent when sending a report.

The platform does not scan any reports for malicious files.

The vendor does not use a specialized malware detection software.

All reports have been validated and triaged by the platform.

Using these assumptions, the framework will be composed of three structural diagrams and one behavioral diagram.

Following the five steps to build a theory in software engineering, these are the defined constructs:

C1: Prevention (system monitoring)

C2: Communication (report opening and downloading procedures)

C3: Coordination (platforms and vendors)

C4: Documentation (allowed file attachments)

C5: Validation (in scope and valid)

C6: Security (malware)

C7: Detectability (malware detection)

C8: Knowledge (file extension and malware relationship familiarity)

Next, are the defined propositions:

P1: The use of report analysis on the report contents improves malware prevention.

P2: The use of report analysis increases security for the vendor.

P3: Report analysis positively affects the validation of reports.

P4: The use of report analysis reduces the potential for vulnerabilities to be exploited from the report content.

P5: The knowledge of malware is reduced without using the report analysis for malware.

P6: The positive effects of report analysis are reduced if there is not enough communication.

P7: Report analysis positively affects coordination between vendor and platform.

P8: Poor report analysis affects documentation for vendor.

The explanations are used to understand why the theory is necessary and important:

E1: By understanding the relationships between file extensions and malware, the safety of viewing reports after malware detection is improved, and the company is protected from malware exposure.

E2: By understanding which attachments each platform allows, a company can choose a platform that has a restricted set of file attachments.

E3: By illustrating the connections between the scopes and constructs, there is an understanding of the reason report analysis for malware is important and what aspects of the bug-bounty life cycle is affected.

Defining the scope has been part of the observational study, but this provides a concise list of the scopes of the theory in terms of validity and interest of each scope.

S1: Technology

- Validity of Scope: The extension and type of malware that are most common and most likely to be allowed as file attachments in reports using over 35 million malware samples.
- Interest of Scope: Of the 95 different extensions used in over 35 million malware samples, the focus is on only the most commonly used file extensions, omitting any extensions that have less than 1000 samples.

S2: Actor

- Validity of Scope: Data is validated by joining and exploring 22 different bug-bounty platforms from an original total of 49 investigated.
- Interest of Scope: Platforms of interest required minimal information needed to register. This information is falsified with minimal effort (i.e. username, email, and name are required).

S3: Activity

- Validity of Scope: Data is validated by a list of allowed attachments for each platform. Within a platform, only accepted file attachments are allowed.
- Interest of Scope: Data on file restrictions. Categorizing file extensions into groups.

S4: Malware Detection

- Validity of Scope: Every malware sample has a reference to the metadata of the sample from VirusTotal. Using VirusTotal helped to classify the type of malware identified in the sample along with the metadata of the malware file.
- Interest of Scope: The relationship between types of malware and file extensions.

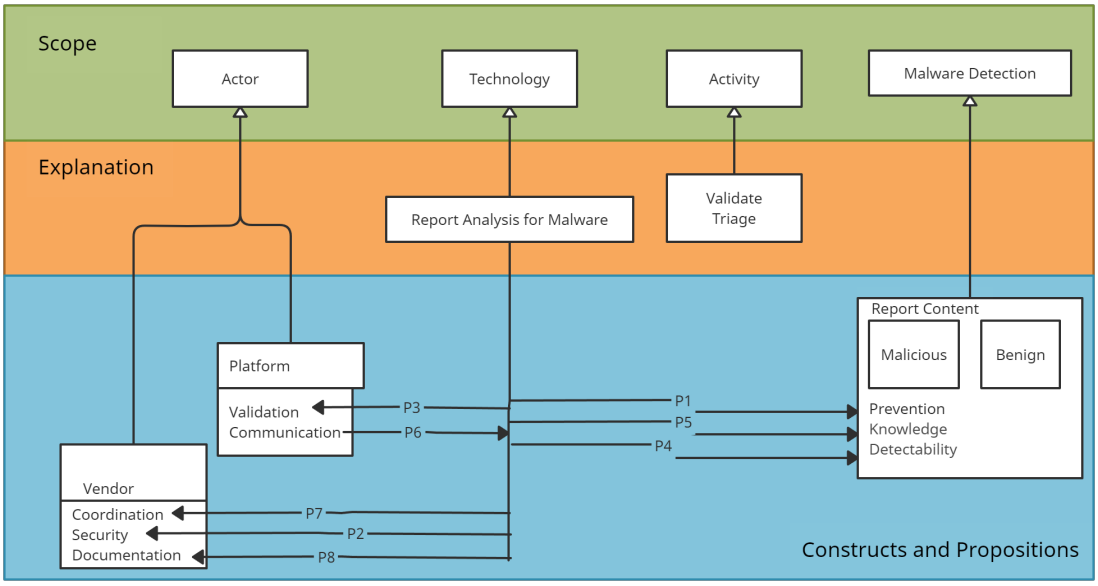

Figure 7.2: Structural: A UML Theory Diagram for the Effects of Malware Analysis in Reports

The final step to building a theory has already been done and will continue to be tested and updated when necessary.

The Relationship Between Bug Report Attachments and Malware

Visuals were created based on the research findings on the relationship between file types and malware types. Figure 7.3 provides the types of malware used by types of files. A complete list of each file extension can be found in Table 6.2. Each type of malware has a different set of files used to attack (different types of malware that have different names

but are still classified under a particular type of malware). In the picture these are called 'subset malware.' Each bug includes a symbol of the malware's actions.

Figure 7.3: Structural: Map of Malware Types with Identified File Types

To understand the importance of this research, a UML class diagram was created to explain the interactions between the components of the framework.

Figure 7.4: Structural: A UML Class Diagram Illustrating the Structure of the Framework

Figure 7.4 shows four components in green. Two of those components have been separated into more detailed objects because a report can contain different types of files and different types of malware. A company has to use a platform(except for the in-house

platforms specified in the background section), so the platform has an input set to 1. Using the platforms from the study, a report can contain zero attachments up to several attachments (some are unlimited). For each report submission, there is only one report submitted, so that connection is set as 1 to 1. Finally, there is a red line that connects the classes Attachments and Malware. This connection used to be 0 to 0, this research changes that '0..0' to '0..*'. Prior to this research, there was never a consideration or procedure in place for detecting malware in bug-bounty reports.

In Figure 7.5, the steps of using the framework are illustrated. It also illustrates how each variable impacts the other variables. The flowchart explains which diagrams are used to lead to the next step.

The vendor is set as the control variable because the vendor (company) does not change. The independent variables are the bug-bounty platforms and the potential malware presented in these findings. They are independent because the platform chooses the file restrictions and the potential malware is based on actual malware samples.

The dependent variables depend on the independent or mediator variables. The report attachment differs based on the prospective bug-bounty platform. The report soundness (legitimacy of the report) depends on whether there was use of malware detection in the report, affecting the potential attack vector on the vendor.

The moderator variable is how the case study affects the knowledge of the vendor; more specifically, this gives the vendor an idea of the types of malware used. This moderator variable can improve malware detection by reducing the types of malware to detect. The mediator variable doesn't affect the report soundness but it enables the understanding of how knowledge of the malware attack vector improves malware detection.

Figure 7.5: Behavioral: Flowchart for Companies Using a Bug-Bounty Platform to Assess Potential Attack Vectors

How to use the Conceptual Framework

The vendor uses the UML class and theory diagrams and a malware diagram to illustrate the structure of this framework. The behavioral diagram is illustrated using the flowchart. The flowchart is used to guide the vendor through the framework.

Conceptual Framework Use Cases

The actor most likely to use this conceptual framework is a company that is looking for a bug-bounty platform to host their program or a company that has a bug-bounty program with a platform and wants to determine the security risks for reports.

First, the vendor finds a potential/current platform. This is done by using Table 5.2 and 5.3 to identify file types and file restrictions. Once the vendor has identified the platform that best fits their needs and has identified the file types, they use the malware diagram (Figure 7.3) to identify their potential malware attacks. The vendor can then choose a malware detection that best fits their potential malware attack. By following these steps, the vendor can increase the report soundness and reduce the malware attack vector.

The framework could also be used to compare platforms to determine which platform in better suited, by iterating through the flowchart for each platform of interest. The reason this is a use case is because not every vendor wants the most secure platform (convenience over security). There is a need for a balance between what file attachments are desired by the vendor and the size of the attack surface the vendor is willing to expose. For example, if the vendor decides that images and compressed files are beneficial report attachments, that means that the vendor could expose themselves to Trojan, Adware, Stego malware, Virus, and PUA/PUP.

CONCLUSION

The results from this research show that if a vendor does not scan reports for malware, there is disastrous potential for malware attacks. For the vendors that don't used malware detection or a virtual environment to view reports, I have compiled a resource connecting malware types to file extensions to help characterize potential attack vectors introduced to the vendor using the report attachments. There are some file types that allow more types of malware than others, but there is a trade-off to consider: malicious attack vectors versus the restrictions on file attachments.

The study also concludes that the standards used by the platform can benefit from adding some type of malware detection clause or advising the use of a detonation chamber[1] for the platform to uphold. However, it is still the responsibility of the vendor to maintain proper security measures that include malware detection for reports.

This research provides a new perspective of where to detect malware and that even when a company opens themselves up to a program to improve their security, there is a need for caution in balancing security with convenience. It can be beneficial to host a program on a platform with no file restrictions, but that decision all depends on the size of the attack surface a vendor is willing to expose.

Future Work

For future work, a thorough study into each type of malware can be done, along with increasing the sample size of the relationship between malware types and file extensions. The list of file types can be expanded into a study of each file extension and its relationship with malware types.

[1] Virtual environment to test for malicious files

In addition, on the researcher's side of the platform's web interface, every vulnerability submission page has a text box that can be used to add a malware script in the report, meaning the malware can embed itself into the report instead of attaching to it. It is worth looking into the impact this has on malware detection.

Also, not all malware is a pure type of malware. Many samples of malware are classified as 'hybrid' malware[2] (i.e. Trojan Virus or Ransom Ad). The interest is in the impact the complex attacks has on the framework.

In addition to hybrid malware, an investigation into Stego malware and its impact on file attachments is warranted since Stego malware is only briefly mentioned in this research. Stego malware is of interest because of its ability to obscure the contents in its file, turning any attack into the accepted file extension used as a report attachment.

Motivation and Contribution

The motivation for this research comes from curiosity, and more specifically, a desire to determine the security of bug-bounty programs for vendors. There was no confirmation on whether vendors could trust that their bug reports do not contain anything malicious, and, therefore, the goal was to see if this confirmation was possible.

Using the framework provides a way for vendors to assess the potential attack vectors, and to understand that there are trade-offs in having restrictions on file attachments for a bug-bounty platform. Using the tables and diagrams to gain knowledge of the relationship between malware and file types, a list of the file types allowed by each selected platform, and a summary of each standards used by the platform, the vendor can proceed with caution to hosting a bug-bounty program and viewing the reported vulnerabilities. My contribution is in providing this framework to help vendors choose a platform that balances the trade-offs

[2] Malware that uses different types of malware to perform a complex attack, an attack that uses multiple attack vectors.

between safety and convenience of report attachments.

This conceptual framework is used to illustrate the importance of understanding the relationship between malware types and file attachments, and understanding the potential attack vectors that accompany a bug-bounty report. Until standards are updated to include a scanning requirement performed by the platform, or the platform advising vendors to use a detonation chamber, vendors must do their part to maintain information security for their customers and services. As long as bug-bounty programs exist, and as long as the platforms are not scanning for malware, the reports are a company's greatest vulnerability.

REFERENCES CITED

[1] Robert S. Anderson, Jacob Benjamin, Virginia L. Wright, Luis Quinones, and Jonathan Paz. Cyber-informed engineering. 2017.

[2] John Aycock. *Spyware and adware*, volume 50. Springer Science & Business Media, 2010.

[3] Lital Badash, Nachiket Tapas, Asaf Nadler, Francesco Longo, and Asaf Shabtai. Blockchain-based bug bounty framework. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 239–248, 2021.

[4] BugCrowd. 2021 ultimate guide to bug bounty, 2021.

[5] BugCrowd. Priority one report 2022: A yea of vulnerabilities in review and a look ahead, 2022.

[6] David M Chess and Steve R White. An undetectable computer virus. In *Proceedings of Virus Bulletin Conference*, volume 5, pages 1–4. Orlando, 2000.

[7] Doina Cosovan, Razvan Benchea, and Dragos Gavrilut. A practical guide for detecting the java script-based malware using hidden markov models and linear classifiers. In *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scienti c Computing*, pages 236–243. IEEE, 2014.

[8] Manuel Egele, Christopher Kruegel, Engin Kirda, Heng Yin, and Dawn Song. Dynamic spyware analysis. 2007.

[9] Ryan Ellis and Yuan Stevens. Bounty everything: Hackers and the making of the global bug marketplace. *Available at SSRN 4009275*, 2022.

[10] CORE 514 Cyber-Informed Engineering. Cyber operations and resilience program. 2021.

[11] Michael Erbschloe et al. Trojan, worms, and spyware: A professional guides to malicious code, 2005.

[12] Mohammad Reza Faghani and Uyen Trang Nugyen. Modeling the propagation of trojan malware in online social networks. *arXiv preprint arXiv:1708.00969*, 2017.

[13] Huw Fryer and Elena Simperl. Web science challenges in researching bug bounties. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 273–277, 2017.

[14] Alex Hoffman and Hal Berghel. Moral hazards in cyber vulnerability markets. *Computer*, 52(12):83–88, 2019.

[15] ISO. Information technology - security techniques -information security management systems - requirements. In *International Standard 27001*, 2013.

[16] ISO. Information technology - security techniques - vulnerability disclosure. In *International Standard 29147*, 2018.

[17] Aron Laszka, Mingyi Zhao, and Jens Grossklags. Banishing misaligned incentives for validating reports in bug-bounty platforms. In *European Symposium on Research in Computer Security*, pages 161–178. Springer, 2016.

[18] Aron Laszka, Mingyi Zhao, Akash Malbari, and Jens Grossklags. The rules of engagement for bug bounty programs. In *International Conference on Financial Cryptography and Data Security*, pages 138–159. Springer, 2018.

[19] Amir Lukach, Ehud Gudes, and Asaf Shabtai. Pua detection based on bundle installer characteristics. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 261–273. Springer, 2020.

[20] Suresh S Malladi and Hemang C Subramanian. Bug bounty programs for cybersecurity: Practices, issues, and recommendations. *IEEE Software*, 37(1):31–39, 2019.

[21] UK NCSC. Secure design principles: Guides for the design cyber secure systems. 2019.

[22] Gavin O'Gorman and Geoff McDonald. *Ransomware: A growing menace*. Symantec Corporation Arizona, AZ, USA, 2012.

[23] Sharon M Ravitch and Matthew Riggan. *Reason & rigor: How conceptual frameworks guide research*. Sage Publications, 2017.

[24] Jukka Ruohonen and Luca Allodi. A bug bounty perspective on the disclosure of web vulnerabilities. *arXiv preprint arXiv:1805.09850*, 2018.

[25] Joanna Rutkowska. Introducing stealth malware taxonomy. *COSEINC Advanced Malware Labs*, pages 1–9, 2006.

[26] Saman Shafigh, Boualem Benatallah, Carlos Rodríguez, and Mortada Al-Banna. Why some bug-bounty vulnerability reports are invalid? study of bug-bounty reports and developing an out-of-scope taxonomy model. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6, 2021.

[27] Forrest Shull, Janice Singer, and Dag IK Sjøberg. *Guide to advanced empirical software engineering*. Springer, 2007.

[28] Amutheezan Sivagnanam, Soodeh Atefi, Afiya Ayman, Jens Grossklags, and Aron Laszka. On the benefits of bug bounty programs: A study of chromium vulnerabilities. In *Workshop on the Economics of Information Security (WEIS)*, volume 10, 2021.

[29] Dag IK Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. In *Guide to advanced empirical software engineering*, pages 312–336. Springer, 2008.

[30] Jessica Sarah Hong Smith. *Sheltering from cyber insecurity? A comparative analysis of New Zealand and Singapore*. PhD thesis, The University of Waikato, 2022.

[31] Tim Stevens, Kevin O'Brien, Richard Overill, Benedic Wilkinson, Tomass Pildegovics, and Steve Hill. Uk active cyber defence: a public good for the private sector. 2019.

[32] Guillermo Suarez-Tangil, Juan E Tapiador, and Pedro Peris-Lopez. Stegomalware: Playing hide and seek with malicious components in smartphone apps. In *International conference on information security and cryptology*, pages 496–515. Springer, 2014.

[33] European Union. Regulation (eu) 2016/679 of the european parliament and of the council. 2016.

[34] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. Goal question metric (gqm) approach. *Encyclopedia of software engineering*, 2002.

[35] Tommy van Steen and Els De Busser. Security by behavioural design: A rapid review. 2021.

[36] Robert K Yin. *Case Study Research: Design and Methods*. sage, 2009.

[37] Mingyi Zhao, Aron Laszka, and Jens Grossklags. Devising effective policies for bug-bounty platforms and security vulnerability discovery. *Journal of Information Policy*, 7(1):372–418, 2017.

[38] Mingyi Zhao, Aron Laszka, Thomas Maillart, and Jens Grossklags. Crowdsourced security vulnerability discovery: Modeling and organizing bug-bounty programs. In *The HCOMP Workshop on Mathematical Foundations of Human Computation, Austin, TX, USA*, 2016.

[39] Jiali Zhou and Kai-Lung Hui. Bug bounty programs, security investment and law enforcement: A security game perspective. In *Workshop on the Economics of Information Security (WEIS). Academic Press. Retrieved from http://hdl. handle. net/1783.1/96436*, 2019.

[40] Aviram Zrahia, Neil Gandal, Sarit Markovich, and Michael Riordan. The effect of an external shock (covid-19) on a crowdsourced "bug bounty platform". 2022.

APPENDICES

APPENDIX A

BUG BOUNTY PLATFORM TABLES

Table 5.3: Third-Party Platforms Report Attachment Restrictions on Size and Count

Company	Location	Max Size	Max Count
Bug Hunt	Brazil	No Limit	10
Bug Base	India	25	No Limit
BugBounter	Turkey	50	1
bugbounty.jp	Japan	5	5
BugBounty.ru	Russia	30	No Limit
BugCrowd	US	50	20
Bugv	Nepal	5	5
Federacy	USA	N/A	N/A
HackenProof	Estonia	50	5
HackerOne	US	250	No Limit
HackRate	Hungary	150	No Limit
HACKTIFY	Hungary	250	No Limit
Hackerfi	Finland	No Limit	1
Immunefi	Virtual	8	20
Inspectiv	US	10	20
Intigriti	Belgium	5	30
Ravro	Iran	100	No Limit
RedStorm	Indonesia	No Limit	No Limit
SafeHats	India	No Limit	No Limit
Safevuln	Vietnam	50	3
SecureBug	Sweden	50	10
WhiteHub	Vietnam	50	10

Table 5.4: Accepted File Types on Bug-Bounty Platforms

Company	Pic	Vid	PDF	Doc	Executable	Compressed	Link
Bug Hunt	p	p	p	p	p	p	
Bug Base	p	p	p			p	
BugBounter	p	p	p				
bugbounty.jp	p						
BugBounty.ru	p	p	p	p	p	p	p
BugCrowd	p	p	p	p	p	p	p
Bugv	p						
Federacy							
HackenProof	p	p	p	p			
HackerOne	p	p	p	p	p	p	p
HackRate	p	p		p			
HACKTIFY	p	p	p	p	p	p	p
Hackerfi	p		p				
Immunefi	p						p
Inspectiv	p	p	p	p	p	p	p
Intigriti	p	p	p	p			p
Ravro	p	p	p	p		p	
RedStorm	p		p	p			
SafeHats	p	p	p	p			
Safevuln	p	p	p	p			
SecureBug	p		p				
WhiteHub	p	p	p	p	p	p	p

The file attachments have specific file types that belong to each category:
Pictures: jpg, fpx, swf, riff, dex, png, gif, bmp
Videos: mp4, mts, m2t
Adobe: pdf
Document: doc, xls, xml, rtf, docx, ppt
Executable: dll, exe, (no extension), pl, py, sh, rb, php Compressed: zip, rar, gz, tar
Links: html, lnk, chm

Table 5.1: Third-Party Platform Data Collection on Registration Information, Geographical Location, Standard Compliance, and Service Options for all Identified Platforms

Platforms	Location	Stnds/Regs	Personal Info	Services
Antihack.me	Singapore	N/A	N/A	BB, PT, RT
BB Carribean	Carribean	ISO-27001	Application	BB
BB Switzerland	Switzerland	GDPR	Application	BB
BugHunt	Brazil	None	Basics	BB, VD
BugBase	India	ISO-27001	Country of Origin	BB, PT, VD
BugBounter	Turkey	GDPR	Basics	BB
bugbounty.jp	Japan	ISO-29147	Basics	BB
bugbounty.sa	Saudi Arabia	N/A	Bank Account, Tele #	BB
bugbounty.ru	Russia	ISO-27001	Basics	BB
BugCrowd	US	FedRAMP	Basics	BB, VD
Bugv	Nepal	None	Basics	BB, PT, VD
CrowdSwarm	UAE	N/A	DoB, Tele #, Addr, Passport	BB, PT
Cyber Army Indonesia	Indonesia	N/A	Indonesia Citizenship	BB, VD
Cyscope	Chile	None	Country	BB
Detectify	Sweden	GDPR	TBD	VD
EpicBounties	Spain LATAM	GDPR	DoB, Addr Passport	BB
Federacy	US	SOC2	Basics	BB, PT, VD
FindBug	Kosovo	GDPR	Complete CTF	BB, VD
Frontal	UAE	N/A	Application	BB, RT, VD
GObugfree	Switzerland	GDPR	Basics	BB, PT
HackenProof	Estonia	GDPR	Basics	BB, PT
HackerOne	US	FedRAMP	Basics	BB, PT, VD
Hackrate	Hungary	GDPR	Basics	BB, PT, VD
HACKTIFY	Hungary	GDPR	Basics	BB, PT, VD
Hackrfi	Finland	GDPR	Basic	BB
Immunefi	N/A	None	Basics	BB
Inspectiv	US	FedRAMP	Basics	VD
Intigriti	Belgium	GDPR	Basics	BB
Open BB	Bangladesh	ISO-29147	Twitter Account	BB
Ravro	Iran	ISO-29147?	Basics	BB
RedStorm	Indonesia	None	Basics	BB, VD
SafeHats	India	ISO-27001	Basics	BB, VD

Continues on next page

Platforms	Location	Stnds/Regs	Personal Info	Services
Safevuln	Vietnam	None	Basics	BB
Secuna	Phillipines	ISO-29147	DoB, Addr, PoI Nationality	BB, PT Audit
SecureBug	Sweden	GDPR	Basics	BB, PT, RT
SlowMist	China		Application	BB, BC, RT
Swarmnetics	Singapore	N/A	Application	BB, PT, VD
Synack	USA	ISO-27001	Application	BB, PT, VD
TestBirds	Netherlands	GDPR	Loc, DoB	BB, PT
TheBugBounty	Malaysia			BB, PT
v1bounty	Germany	GDPR	Basics	BB, VD
Vulbox	China	None	Tele#(+86)	BB, VD
Vulnerability Lab	Germany	GDPR	Basics	BB
Vulnscope	Chile		Country, DoB	BB, PT
WhiteHub	Vietnam	None	Basics	BB, PT
YesWeHack	France	GDPR		BB, VD
Yogosha	France	GDPR	Community	BB
Zerocopter	Netherlands	GDPR		BB, VD

Table 5.2: Selected Third-Party Platforms with Geographical Location, Standards Compliance, and Year Established

Company	Location	Year Est.	Compliance
BugHunt	Brazil	2020	None
BugBase	India	2021	29147
BugBounter	Turkey	2020	GDPR
BugBounty.jp	Japan	2015	ISO-29147
BugBounty.ru	Russia	2021	ISO-27001
BugCrowd	US	2012	FedRAMP
bugv	Nepal	2020	None
Federacy	USA	2018	SOC2
HackenProof	Estonia	2017	GDPR
HackerOne	US	2012	FedRAMP
HackRate	Hungary	2020	GDPR
Hackerfi	Finland	2016	GDPR
HACKTIFY	Hungary	2020	GDPR
Immunefi	Virtual	2020	GDPR
Inspectiv	US	2018	FedRAMP
Intigriti	Belgium	2016	GDPR
Ravro	Iran	2019	N/A
RedStorm	Singapore	2018	None
SafeHats	India	2012	ISO-27001
SafeVuln	Vietnam	2014	None
SecureBug	Sweden	2019	GDPR
WhiteHub	Vietnam	2019	None

Table A.1: Bug-Bounty Platform Resources

Company	Weblink
BountySource	https://bountysource.com
BugHunt	https://www.bughunt.com.br/index.html
BugBase	https://bugbase.in/
BugBounter	https://bugbounter.com/
Bug Bounty Caribbean	https://bugbountycaribbean.com/
BugBounty.jp	https://bugbounty.jp/
BugBounty.ru	https://bugbounty.ru
BugBounty.sa	https://bugbounty.sa/
Bug Bounty Switzerland	https://bugbounty.ch/
BugCrowd	https://www.bugcrowd.com/
bugv	https://bugv.io/
Cobalt	https://www.cobalt.io/
CrowdSwarm	https://www.crowdswarm.io/
Cyber Army Indonesia	https://www.cyberarmy.id/en
Cyscope	https://cyscope.ch
Detectify	https://detectify.com/
Epic Bounties	https://www.epicbounties.com/
Federacy	https://federacy.com
FindBug	https://findbug.io/
Frontal	https:frontal.io
GOBugFree	https://gobugfree.com/
HackenProof	https://hackenproof.com/
HackerOne	https://www.hackerone.com/
HackRate	https://hckrt.com/
Hackrfi	https://hackr.fi/
HACKTIFY	https://www.hacktify.eu/en/home/
Immunefi	https://immunefi.com/
Inspectiv	https://www.inspectiv.com/
Intigriti	https://www.intigriti.com/
Open Bug Bounty	https://www.openbugbounty.org//
Ravro	https://www.ravro.ir/
RedStorm	https://www.redstorm.io/
SafeHats	https://safehats.com/
SecureBug	https://securebug.se/
SlowMist	https://www.slowmist.com/
Swarmnetics	https://www.swarmnetics.com/

Continues on next page

Company	Weblink
Synack	http://synack.com/
TestBirds	https://testbirds.com/en/
The Bug Bounty	https://thebugbounty.com/
V1 Bug Bounty	https://v1bounty.com/
VulBox	https://vulbox.com/
Vulnerability Lab	https://vulnerability-lab.com/
Vulnscope	https://vulnscope.com/
SafeVuln Viettel	https://safevuln.com/
WhiteHub	https://whitehub.net/
YesWeHack	https://yeswehack.com/
Yogosha	https://yogosha.com/
Zero Day Initiative	https://zerodayinitiative.com/
Zerocopter	https://zerocopter.com/

APPENDIX B

MALWARE METADATA

Table 6.1: Data Collection of File Extension and Count on Malware Samples with More than 20 Samples per Extension

Extension	Count	Extension	Count	Extension	Count
exe	18318271	html	10175562	(no extension)	2813930
dll	2180563	zip	690314	pdf	291078
rar	246302	m2t	183796	gz	82844
jpg	77256	doc	71797	dex	54009
lnk	29421	png	28290	xml	20309
gif	18915	php	18192	xls	17697
fpx	16102	xlsb	12898	swf	11072
sh	4972	pl	4797	rtf	4311
mp3	3619	bmp	3407	chm	2366
docm	2354	xlsm	1827	mp4	1691
riff	1310	mts	1072	asf	836
bz2	762	plist	662	py	645
wmf	608	docx	595	tar	582
ttf	512	ogg	462	xlsx	417
ppt	415	wav	356	tif	339
svg	229	(386)	207	a	207
torrent	145	mov	110	rsrc	101
eps	92	dotm	76	m4a	59
rb	55	xlam	53	webp	49
otf	43	mpg	31	pptx	26
webm	23	acr	21		

Table 6.2: Selected File Extensions with File Count and Category(Type)

Extension	Count	Category	Extension	Count	Category
exe	18318271	executable	html	10175562	link
(no extension)	2813930	executable	dll	2180563	executable
zip	690314	compressed	pdf	291078	Adobe
rar	246302	compressed	m2t	183796	video
gz	82844	compressed	jpg	77256	image
doc	71797	document	dex	54009	image
lnk	29421	link	png	28290	image
xml	20309	document	gif	18915	image
php	18192	script	xls	17697	document
fpx	16102	image	xlsb	12898	document
swf	11072	image	sh	4972	script
pl	4797	script	rtf	4311	document
mp3	3619	audio	bmp	3407	image
chm	2366	link	docm	2354	document
xlsm	1827	document	mp4	1691	video
riff	1310	image	mts	1072	video
py	645	script	docx	595	document
tar	582	compressed	ppt	415	document
mov	110	video	rb	55	script

Table 6.3: Relationship between Types of Malware and File Extensions

File Ext	Ad	PUA/P	Spy	Script	Trojan	Ransom	Virus	Worm
no ext					p		p	
bmp						p		
chm					p			
dex	p				p		p	p
dll					p			
doc						p		
docx	p	p	p		p			
exe					p			
fpx					p			
gif	p		p		p			
gz			p	p	p			p
html					p			
jpg					p			
lnk								
m2t			p		p			
mp3					p			
pdf				p		p		
php				p	p			
pl			p		p			
png			p		p			
ppt				p				
py	p	p			p			p
rar				p				
rb								
riff					p			
rtf			p	p	p			p
sh					p			
swf								
tar								
xls					p			
xml	p				p	p	p	p
zip								

APPENDIX C

MALWARE/FILE TYPE ANALYSIS DATA

Table 3.1: Malware File Extension Used in Study

Extension	File Description
(no extension)	Linux-Dev86 executable headerless
asm	assembler source ASCII text
asm	assembler source Non-ISO extended-ASCII text
asf	Microsoft ASF
elf	ELF 32-bit LSB executable Intel 80386 version 1 (SYSV)
dos	DOS batch file ASCII text with CRLF line terminators
dos	DOS batch file ISO-8859 text
dos	DOS executable (COM)
c	C source ASCII text
com	COM executable for DOS
com	COM executable for MS-DOS
dat	data
dex	Dalvik dex file version 035
dll	PE32 executable (DLL) (console) Intel 80386
dll	PE32 executable (DLL) (GUI) Intel 80386 for MS Windows
dll	PE32 executable (DLL) (native) Intel 80386 for MS Windows
dll	PE32+ executable (DLL) (GUI) x86-64 for MS Windows
doc	Composite Document File V2 Document Can't read SSAT
doc	Composite Document File V2 Document Little Endian Os
exe	PE32 executable (console) Intel 80386 for MS Windows
exe	PE32 executable (GUI) Intel 80386 for MS Windows
exe	PE32 executable (native) Intel 80386 for MS Windows
exe	PE32 executable Intel 80386 for MS Windows
exe	PE32+ executable (GUI) x86-64 for MS Windows
gif	GIF image data version 89a 10 x 12
gz	gzip compressed data from NTFS filesystem (NT)
gz	gzip compressed data from Unix
html	HTML document ASCII text
html	HTML document ISO-8859 text
html	HTML document Non-ISO extended-ASCII text
html	HTML document UTF-8 Unicode (with BOM)
img	DOS/MBR boot sector code offset 0x3c+2 OEM-ID "MSDOS5.0"
iso	ISO-8859 text with CRLF line terminators
iso	Non-ISO extended-ASCII text
jar	Java archive data (JAR)
java	compiled Java class data
jpg	JPEG image data JFIF standard 1.01 aspect ratio density 1x1

Continues on next page

Extension	File Description
ms	MS Windows shortcut Item
pas	Pascal source ASCII text
pdf	PDF document version 1.3
php	PHP script ASCII text with CR line terminators
pl	perl script ASCII text executable
pl	perl script executable (binary data)
png	PNG image data 8-bit/color RGBA non-interlaced
rar	RAR archive data
rtf	Rich Text Format
smgl	exported SGML document ASCII text
smtp	SMTP mail ASCII text with CRLF line terminators
swf	Macromedia Flash data (compressed)
txt	ASCII text
txt	UTF-8 Unicode text
xls	Composite Document File V2 Document Little Endian Os 0
xml	XML document text
zip	Zip archive data

Table C.1: Malware and Corresponding File Types

Malware	Pics	Docs	Executable	PDF	Compressed	Links	Videos
Adware			p		p		
PUA/PUP	p	p	p	p	p	p	p
Stego		p	p				
Ransomware			p				
Script-Based			p			p	
Spyware	p		p	p	p	p	
Trojans			p		p		
Virus			p			p	
Worms							

APPENDIX D

COMPONENTS OF THE FRAMEWORK

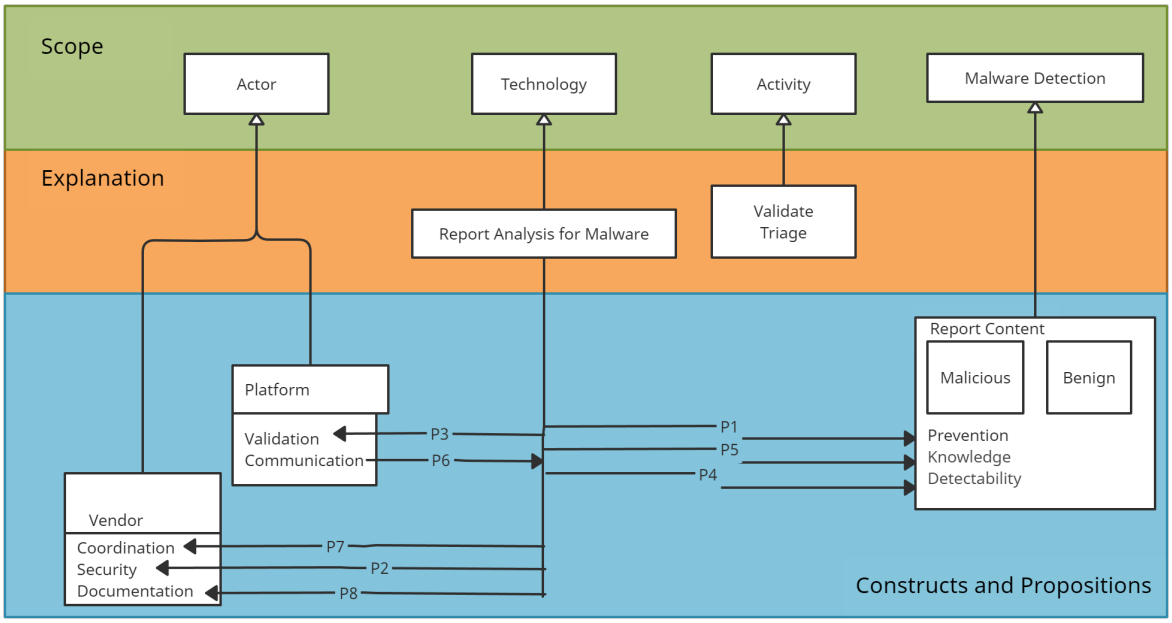

Figure 7.2: Structural: A UML Theory Diagram for the Effects of Malware Analysis in Reports

Figure 7.4: Structural: A UML Class Diagram Illustrating the Structure of the Framework

Figure 7.3: Structural: Map of Malware Types with Identified File Types

Figure 7.5: Behavioral: Flowchart for Companies Using a Bug-Bounty Platform to Assess Potential Attack Vectors