BALKAN JOURNAL OF ELECTRICAL & COMPUTER ENGINEERING, Vol. 12, No. 4, December 2024

Research Article

Analysis of Malicious Files Gathering via Honeypot Trap System and Benchmark of Anti-Virus Software

Melike Baser, Ebu Yusuf Guven, Muhammed Ali Aydin

***Abstract*—In the age of widespread digital integration, the rise in cyber threats is evident. Cyber attackers use malicious software (malware) to compromise data and exploit system resources, employing tactics such as remote control or ransom through data encryption. Despite the common use of antivirus software with signature-based detection, this study reveals its limitations. Using a honeypot trap system on Google Cloud, suspicious files uploaded by attackers were analyzed. Results from evaluating these files with 64 antivirus programs show that relying solely on signature-based methods is insufficient. Only three programs had success rates exceeding 90%, while the majority had success rates predominantly below 70%. This underscores the need for diverse detection techniques alongside signature-based methods to enhance cybersecurity. The repository containing collected malicious files and the Python script is available on Github, serving as a valuable research resource for further exploration.**

***Index Terms*—Malware, Cowrie, Trap System, Honeypot, Anti-Virus.**

I. Introduction

IN the contemporary era, the pace of development in both the internet and information systems continues to increase steadily, consequently leading to a proportional escalation in the risks associated with cyber attacks.Malicious software, or malware, is the primary instrument used by cybercriminals to carry out targeted assaults and evade security protocols [1]. Malware developed by cyber attackers can be considered among the severe attack vectors that compromise the security of systems. In addition to the scientific and technical challenges of malware analysis, it is also complicated to obtain malware safely. In order to evaluate suspicious files, signature-based, behavior-based transactions from movements on the system by passing through the summary functions of the files and static analysis operations can be performed over the source code. One of the ways to obtain malware is to collect malware that infects systems. The collected malware should be investigated through open sources. The use of trap systems called Honeypot [2] which attract attackers to find malware used in current attacks and make them think that they have gained access to a real system is also preferred by researchers.

According to the 2022 statistics of Cyber Attacks [3], it seems that malware is developing and becoming more sophisticated than ever. Malware [4], is software designed for stealing important information and gaining access to the remote target system or damaging system resources. Malware comes in many forms, including viruses, worms, trojan horses, rootkits, backdoors, botnets, spyware, and adware [5]. Malware can be seen as a significant security threat facing end-users today. Under the current conditions, it is almost impossible to prevent infection with viruses since it is unnecessary to use computer networks, flash drives, pirated software, and more. Statistics show that every computer is infected with at least one virus [6].

Anti-virus applications detect and classify malware by constantly scanning files and comparing their signatures with known malware signatures. Malware signatures are usually generated by anti-virus experts who examine the malware samples collected. These malware signatures can be file names, text strings, or regular expressions of byte code. Signature-based methods can only detect malware found in databases that have not changed significantly. According to the Pyramid of Pain, changing the hash information is considered the footprint where attackers spend the least amount of effort and time [7]. Accordingly, malware can hide its malicious signature using code obfuscation techniques that make the code look quite different from the original version, making the reliability of signature-based systems questionable.

Within the scope of this study, suspicious files collected through Cowrie, a trap system, were checked based on signatures by using anti-virus applications on Virus Total. Furthermore, a Python script has been developed that checks the databases of other anti-virus software via the Virus Total API for signature checking suspicious files uploaded by attackers on Honeypot. Our goal is to determine which antivirus applications

Melike BAŞER is with the Department of Computer Engineering, Engineering Faculty, Istanbul University - Cerrahpasa, Istanbul, TURKEY e-mail: melike.baser@iuc.edu.tr

Ebu Yusuf GÜVEN is with the Department of Computer Engineering, Engineering Faculty, Istanbul, TURKEY e-mail: eyguven@iuc.edu.tr

Muhammed Ali AYDIN is with the Department of Computer Engineering, Engineering Faculty, Istanbul University - Cerrahpasa, Istanbul, TURKEY e-mail: aydinali@iuc.edu.tr

Manuscript received Jul 1, 2024; accepted Nov 22, 2024
doi: 10.17694/bajece.1506554

are able to recognize the signatures of recent suspicious files uploaded to our trap system by attackers at the time of upload. In this study, we observed that some signature-based anti-viruses were able to recognize the vast majority of recent suspicious files, while signature-based methods were expected to fail. Signature-based anti-viruses are especially preferred in the field because they offer the advantages of fast response and low cost. In our work, we developed a Python application that works on file signatures to quickly evaluate malware uploaded by attackers to the honeypot environment. The types of suspicious files identified as malicious files and which malware family they belong to have been examined and classified. In addition, the returns of anti-virus applications to suspicious files were evaluated, and anti-virus applications were compared among themselves. The main contributions of this study are:

- Introducing a new data set of malicious files[1],
- Detection of files uploaded to Cowrie Honeypot using Virus Total,
- Classification of malware using signature-based detection and analysis of its statistics,
- Determining the success of signature-based detection methods,
- Evaluation and comparison of anti-virus applications,
- Evaluation of the working mechanism of anti-virus applications and discussion of the adequacy of the signature-based detection method.

The rest of this article is organized as follows. In Chapter 2, the relevant studies are described. Section 3 presents malware classification, detection, and analysis methods. Chapter 4 includes the experimental processes of the work done, while Chapter 5 conveys the results obtained from this process. Finally, the results obtained in Chapter 6 were discussed, and the study was terminated by making the necessary recommendations.

II. Related Work

Malware; their goals, techniques, and methods are cyber threat vectors that change and develop daily. Anti-virus applications that use different methods such as file signatures, source code, and behavioral analysis to protect against malware have emerged. Anti-virus software is the biggest obstacle to malware, such as damaging software and hardware, stealing information, using it for its purposes, or demanding a ransom. Malware that uses many methods to circumvent anti-virus applications has reached our day by undergoing substantial changes [8]. In the face of increasing and changing threat vectors, cyber security researchers have also needed to improve how they detect malware [9]. Researchers have classified malware according to its change and development, its goals and objectives, and the vulnerabilities it exploits. While the first type of malware developed was known to be a joke virus, on the other hand, the Cryptojacking malware, which produces crypto money by exploiting device resources, appeared [10], [11]. While there are fundamental classifications such as virus, worm, trojan horse, and ransomware in some sources, some studies include more detailed classifications such as a rootkit, botnet, and fileless malware [12], [13], [14].

Malware detection methods can generally be classified into three categories: signature-based, behavior-based, and heuristic-based. Tibra Alsmadi and his colleagues classified it as signature-based detection, heuristic-based detection, and sandbox detection [15]. They also explained the advantages and disadvantages of each when making this classification. Anti-virus applications installed on computer systems usually use a Signature-based detection technique and form the basis of security software [16]. Rohith et al. shared how an anti-virus program detects a suspicious file, how to prevent the proliferation of malware on computer systems, and methods for detecting the identity of suspicious files [6].

Although anti-virus software effectively blocks known malware threats, it creates portability, resource consumption, and cost problems. It is necessary to install a separate anti-virus application on each computer, and also, anti-virus applications have periodic needs, such as updating and license renewal. Akshay Chavan and his colleagues have designed an anti-virus that can be found on any portable device and scan the computer system for signature-based and heuristic analysis when connected [16]. Considering that a portable anti-virus program uses a static database and thousands of brand-new viruses are developed customary, it includes the various risks. Over the Internet, having a dynamic database is becoming crucial. The study that designed another anti-virus belongs to Botacin et al. The framework for the proposed HEAVEN model Intel x86/x86-64 and MS Windows was introduced to improve the performance and effectiveness of Anti-virus. The HEAVEN model detected almost all malware with a data set consisting of 10,000 malware and 1,000 benign software samples from different categories [17].

With the development of Artificial Intelligence technologies, studies using artificial intelligence in malware detection are also increasing. By providing a detailed review of ways to improve the efficiency of existing malware detection technologies, Faruk and his colleagues have shown that artificial intelligence is promising for developing anti-malware systems [18]. Various artificial intelligence methods used in conjunction with the signature-based detection technique have become one of the principal methodologies in malware detection. Lima et al. classified the malware with an average accuracy of %98.32 using the model they created with Artificial Neural Network in anti-virus

[1]https://github.com/istec-iuc/cowrie-logs

applications [19]. It has been shown to give statistically superior and more effective results than the signature-based Anti-viruses. Rani et al. also applied the K-Nearest Neighbors algorithm to detect anomalies and discussed the challenges associated with implementing malware classifiers [20]. Intrusion detection systems developed using multiple Machine Learning methods have provided greater security against malware [21]. Assegie demonstrated a signature-based malware detection model based on classification and analysis using the K-Nearest Neighbors (KNN) model and the application programming interface (API) on a Kaggle dataset with %98.17 accuracy. In addition, the grid search method was used to find the optimal K value to obtain higher classification accuracy [22]. With the advancement of smartphone technology, the development of mobile applications has increased rapidly, and new challenges have emerged, such as the inability to distinguish between benign and malicious malware applications. Mo'ath Zyout et al. present Conv1d and LSTM methods for classifying mobile applications as benign or malicious based on Android permissions, achieving high precision and accuracy (98.16%, 97.72%, and 96.63%, 96.69%, respectively) on the CICMalDroid 2020 dataset. Conv1d, specifically in binary classification, outperformed the LSTM model when compared with the Mal-Prem dataset [23].

In order to perform the detection and analysis of malicious software, first of all, data consisting of malware is needed. Although examples are obtained from malware attacks that infect systems and come to institutions and organizations, it is impossible to know the newly developed types before infecting a system. Newly developed species that come from the same malware family and are considered a different variants can be discovered using honeypot systems. Detection and analysis of suspicious files uploaded to honeypots can allow improving the measures that can be taken against malware. Sethia et al. classified the malware they collected through Honeypot according to the protocols they received attacks from, and it was seen that the most attacks came from SQL 2000XP, SMB, and FTP protocols [24]. The fact that protocols frequently used in corporate structures is attacked shows that malware targets organizations. Matin et al. proposed a framework for collecting Portable Executable malware using the Modern Honey Network. The Honeypot used is the Modern Honey Network with Dionaea sensor. They classified the types of malware using Virus Total [25]. Kyriakou et al. introduced a distributed honeypot system that records links to the commonly used protocol, detects malware using Virus Total's analysis engine, and works in a distributed manner [26]. The study showed that in addition to assisting in an organization's risk assessment, real-time statistics and analysis can be provided against honeypots and attackers. Chris Moore examines how honeypot policies can be used to detect and mitigate a ransomware attack on the Microsoft Windows network [27]. Wang and his colleagues extensively analyzed the similarity of malware binaries caught by IoTCMal, which they proposed as IoT Honeypot, and discovered eight malware families controlled by at least 11 groups of attackers [28]. Research studies show that honeypots are valuable for obtaining and analyzing malware.

In this investigation, an evaluation study was carried out on the malware files collected with Cowrie Honeypot installed on Google Cloud. Within the scope of the study, a signature-based detection technique was used, and a program script was developed that could evaluate the attacks on Cowrie instantly through Virus Total. Responses from Anti-virus applications running on Virus Total distinguish suspicious files from malware or benign files. The success of signature-based methods against files used by attackers to attack current systems was demonstrated. Statistics were determined by classifying malware, and the performance of anti-virus applications was evaluated.

III. Malware

Malware is software that aims to perform an unauthorized operation that will have a negative impact on the confidentiality, integrity, or usability of the information system [29]. Detection of malicious software is divided according to how the developed software works and the methods of analysis.

A. Malware Types

Malware has variations depending on the development purpose, the way it works, or the targeted attack. These types of malware are:

a) Virus: It is a malicious program that enters the system without the user's knowledge or agreement and can duplicate itself by activating through an application file. A virus can corrupt or delete data on a computer, propagate to other machines via email, and destroy the entire hard drive [29], [30].

b) Worm: It is a self-replicating and propagating program that uses network mechanisms to propagate itself [30]. The critical difference between viruses and worms is that for a virus to start replicating requires human intervention, such as opening the infected file. In contrast, worms can reproduce without intervention [31].

c) Trojan Horse: A computer program with a covert and potentially malicious function that appears to have a useful function but escapes security mechanisms by exploiting the legitimate powers of a system entity [32].

d) Spyware: It is malicious software that gathers data without the user's knowledge or consent [32]. It may do various tasks, including recording the keys that a user presses on the keyboard, tracking the websites visited, scanning the data on the hard drive, and keeping track of Internet searches. They do not copy themselves in any way from one machine to another, unlike viruses [33].

e) Adware: It refers to a particular class of malware that uses the computer or other device to display unwanted adverts. While adware can occasionally be safe, some pop-ups also aim to gather data and information through targeted advertisements in addition to displaying advertisements [33]. It can use numerous advertising links to route you to malicious websites and infected pages.

f) Rootkit: It is a malicious program designed for malicious people to gain privileged access to devices. Rootkits cannot reproduce by themselves and do not pose a danger to the system [34]. However, they can often be part of more sophisticated threats, and attacks, such as malware installation.

g) Botnet: Computer networks infected with malware are referred to as botnets. Bot refers to remotely controllable software, applications, or code script installed in machines [35].

h) Ransomware: It is a type of malware that encrypts the files of the targeted person or organization. With this attack, the cyber threat actor prevents the victim from accessing their data until the victim pays the ransom [36]. There are two types of ransomware. The first is locker ransomware (Locker ransomware), and the second is encrypted ransomware (Crypto ransomware). Locker ransomware affects basic computer functions, while Encrypted ransomware encrypts certain files [37].

i) Backdoor: They are applications that allow cybercriminals or attackers to access computers remotely[38]. Backdoors can also spread via malicious apps on mobile and smart devices.

j) Downloader: It is a type of Trojan horse that downloads other malware onto a computer. Also called a dropper [39]. The downloader must be connected to the Internet to be able to download the files.

k) Cryptojacking: It is a type of malware in which victims' devices (computers, smartphones, tablets, and even servers) are used secretly and without permission to mine cryptocurrencies without their knowledge[10]. Unlike other threats, cryptojacking is designed to be hidden entirely from the victim.

l) Fileless Malware: Instead of downloading or installing files to infect a system, they are software that infects the system via utilizing existing software (WMI, Powershell, etc.) on the device. It can get through security precautions, including signature detection, hardware verification, pattern analysis, and timestamping, and generally leaves no trace. Since it does not leave a trail, it can get past ”Signature-Based Antivirus” systems[10].

B. Malware Detection Techniques

There are fundamentally two techniques used to detect malware. One of these techniques is Signature-Based Detection, and the other is Behavioral Based Detection. Apart from these techniques, techniques developed according to the types of harmful files or the purpose of the studies are also used. Therefore, this study classified malware detection methods into three different categories.

a) Signature-based Detection: It is the most widely used malware detection technique. A signature is a sequence of bytes used to identify specific malware. Signature-based anti-virus software maintains a repository of known malware signatures, which is updated as new threats are discovered. Signature-based detection is simple, fast, and effective against common types of malware. The most crucial disadvantage is that it requires an up-to-date signature database. Malware that is not in the database cannot be identified and detected without a current signature database. In order to avoid signature detection, several obfuscation techniques are employed. Using these methods, malware cannot be found [40].

b) Behavioral-based Detection: The behavior-based malware detection technique observes program behavior with monitoring tools and determines whether the program is malicious or benign. Even if the program codes are changed, the behavior of the program will be similar; therefore, most of the new malware can be detected by this technique. [41]. Various Virtual Machine and Sandbox tools can be used in the Behavioral-based detection technique.

c) Heuristic-based Detection: It is a detection method that evaluates the features extracted from a software file and its behavior on the system (memory, processor, disk, etc.) with artificial intelligence methods [42]. With artificial intelligence techniques, the approach can use the outputs of signature-based and behavior-based analysis methods as features. As a result, the model developed according to the applied artificial intelligence method can detect unlearned malware that has not been seen before.

C. Malware Analysis Methods

Dynamic analysis and static analysis are the two main methods used to analyze malware. In specific research, these are supplemented by using hybrid analysis and memory analysis approaches [43], [44].

a) Static Analysis: Static analysis is the technique of analyzing the suspicious file with methods such as API call, Opcode, and N-gram without running it. The analysis method is generally used in the first stage, enabling the correct decision on how to classify or analyze the suspicious file and which direction the following analysis studies should focus on [45].

b) Dynamic Analysis: Dynamic analysis refers to the process of analyzing the changes on the network, memory, processor, RAM, or applications in the system by running a code or script by observing with tools such as Wireshark, Autoruns, and Sandbox. The purpose of dynamic analysis is to detect malicious activity by the executable while running without compromising the security of the analysis platform [46].

c) Hybrid Analysis: The hybrid analysis technique looks at behavioral parameters such as consuming processing power, changes in memory, and applications that affect system performance or disrupt its operation to analyze any malware signature and improve malware analysis. This method, which is used to overcome the shortcomings of static and dynamic analysis techniques, increases the ability to accurately detect malware [47].

d) Memory Analysis: Memory analysis has become an effective technique in malware analysis. Memory analysis uses the memory image to analyze information about running programs, the operating system, and the overall state of the computer. Investigations consist of two steps memory acquisition and memory analysis. In memory collection, the target machine's memory is freed using tools such as Memoryze, FastDump, and DumpIt. In memory analysis, memory images are analyzed using tools such as Volatility and Rekall [48].

IV. Experimental Method

Malware is software that disrupts or blocks the operation of systems and is designed and analyzed for attack purposes. In our study, malware files collected with the Cowrie honeypot system we built on Google Cloud were compared and evaluated with antivirus applications using signature-based detection method. Signature-based antiviruses are widely used in the field, offer the advantages of fast response and low cost, and are simple and effective. For this purpose, a Python script was developed to analyze 50 malicious files obtained from the Cowrie honeypot on Virus Total.

A. Preparation Process

Malware is one of the methods used by attackers to disrupt the accessibility, integrity, and confidentiality of the system. Therefore, analysis of attacks on systems by malicious software is essential to ensure security by taking precautions. The motivation behind catching and analyzing malware is to study its mechanism and how it interacts with the operating system. By understanding what types of activities are taking place over the network, he can develop a security mechanism to protect people and various organizations from such malware attacks. In this study, a tool that acts as a trap is used to attract attackers and make them think that they have gained access to a real system, and it is evaluated whether the files obtained from the attacks on this trap system are harmful. As a result of the evaluation, anti-virus applications were also compared.

In our study, the environment where the trap system was configured was preferred as Google Cloud Platform. Google Cloud Platform is a cloud information platform that provides server infrastructure services to the end-user and allows creating and running Virtual Machines [2]. Google Cloud Platform has been preferred in terms of ease of use by giving an external IP address to the virtual machine installed on it and allowing the machine to change the firewall rules easily. A virtual machine with 2 CPUs, 4 GB Memory features, and Ubuntu 18.04 LTS operating system is configured on the Google Cloud Platform. For this virtual machine, the Americas region has been chosen as the location. All incoming network traffic (ingress) is allowed by updating the firewall rules.

Cowrie honeypot system was installed on the virtual machine on Google Cloud Platform. Cowrie is a low-medium interactive SSH and Telnet honeypot system that can record brute-force attacks and the attacker's shell interaction [3]. The findings are based on data collected from the virtual machine running on the Google Cloud Platform between when the Cowrie honeypot trap system was first activated and the last date it was examined. As a result, over the duration of its 47-day longevity, the honeypot system received 3,626,394 attacks, all of which were recorded in log files. In addition, the honeypot system also kept a record of the suspicious files that were collected throughout the attacks[49].

B. Malware Gathering

Cowrie honeypot is a decoy system that simulates SSH and Telnet protocols, allowing attackers to perform various activities within the system via remote connection. In this system, 50 suspicious files were obtained as a result of attacks carried out over FTP (File Transfer Protocol), which is used by attackers to upload and download files. While the attacker activities in the Trap System are recorded under the var/log folder, a different directory is used for uploaded files. Files uploaded by attackers via FTP are stored under the var/lib/downloads directory and 50 suspicious files stored in this directory were retrieved from the system for further analysis. 64 different anti-virus applications were analyzed and an automatic analysis script was developed in Python to evaluate the files. With this script, the results from various anti-virus applications were compared and the effectiveness of the Signature Based Detection technique was discussed. The generated dataset contains the tags of files flagged as suspicious by anti-virus applications and is intended to classify various types of malware. Specifically, the dataset contains different malware categories such as Trojan, Mirai, Backdoor, Malware, CoinMiner, Downloader, and Other, with each category containing types and variants of files designed to perform malicious activities. For example, Trojan and Backdoor files provide unauthorized access to the system, while Mirai files are used to carry out botnet attacks against IoT

[2]https://cloud.google.com/why-google-cloud

[3]https://cowrie.readthedocs.io/en/latest/

devices. This dataset is an important resource for cybersecurity researchers and threat analysts, enabling in-depth analysis of malware identification and classification. When the files are analyzed in general, it is seen that 50 files have a total size of 52 MB. In this case, there is an average of 1 MB per file, with the largest file being 3024 KB and the smallest file being 2 KB. The content of most of the files is unreadable due to obfuscation, while the readable files usually contain bash script or HTML code.

C. Evaluation of Malware

In order to find and analyze malware, various techniques are used. The signature-based detection approach is the quickest and most famous of these approaches. The signature-based detection approach uses tools like Virus Total, Trend Micro, and URLVoid. Virus Total [4] is a free tool with a desktop downloadable or online API for analyzing suspicious files, hashes, or URLs. It uses a set of Antivirus engines to facilitate the detection of various malware [50]. Apps are neither classified as malicious nor benign by VirusTotal. About 60 antiviral scanners are used to determine the content and runtime of an application or executable file based on its hash. As a result, the platform user must choose how to interpret this information to determine if a file is malicious. There are no standard procedures for interpreting scan results from Virus Total to directly tag applications [51]. In this study, malicious files were detected using the signature-based detection method with Virus Total's assist, and the anti-virus programs' responses were evaluated.

D. Control Application

The files uploaded to the system by the attackers during the time the Cowrie Honeypot was attacked were recorded. Various anti-virus programs on Virus Total, which use a signature-based detection method, have been given suspicious files. The files were automatically uploaded to Virus Total thanks to the developed Python script, and the responses provided by the anti-virus software were gathered. The Figure 1 depicts the process's flow. On Github, malicious files and Python script are shared [5].

V. Result

Cowrie Honeypot, which we installed on Google Cloud, tracked the attacker's behavior for 47 days and analyzed 50 suspicious files that the attackers uploaded to the system. The hashes of suspicious files with the Python script we developed were compared over the Virus Total API (Application Programming Interface), and anti-virus applications' malware signature detection results were evaluated. Suspicious files are classified as 'harmful', 'not harmful', or 'not responding' as described in the Table I. Sixty-four anti-virus apps on Virus Total identified files as 'positive'; files it does not identify as harmful are 'negative'; it also returned 'not responding' for files where it could not specify any class. Apart from these, there were also Anti-virus applications that did not respond within a certain period. According to the results indicated in the Figure 2, %34 of the 64 Anti-virus applications received positive feedback, which meant that there were malicious files, while %58 of the negative return was received, which resulted in the uploaded files that were not harmful. Anti-virus applications that cannot classify these files as harmful or not harmful also appear in %8.

[4]https://www.virustotal.com/gui/home/upload
[5]https://github.com/istec-iuc/cowrie-logs

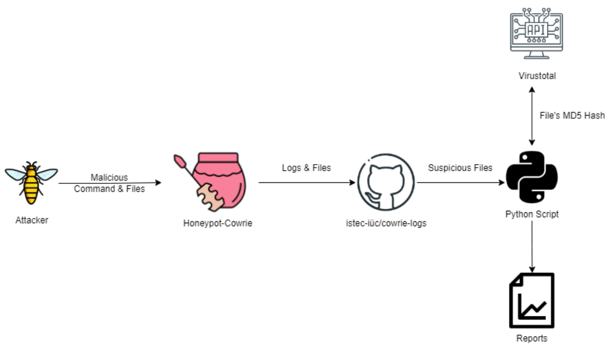


Fig. 1: Detection and evaluation of suspicious files obtained

TABLE I: Descriptions of antivirus responses to suspicious files

State	Meaning
Positive	The file's signature was found in the database of the Anti-virus application, and it was determined that it is malware.
Negative	The database of the anti-virus application did not find the file's signature and was found to be free of malware.
Not Responding	The file's signature could not be found in the database of the anti-virus application and could not respond in the required time.

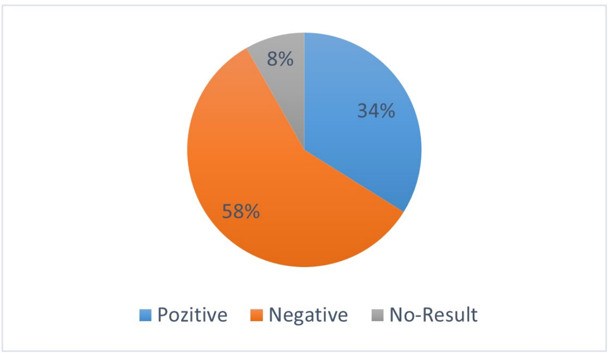


Fig. 2: Evaluation result percentages of Anti-virus applications in malware detection

Malicious files are divided into several classes according to their purpose, mode of operation, or different parameters. These have been extensively mentioned under the heading 'Malware Types'. According to the answers received from the Anti-virus applications by performing the analysis, six main headings were determined within the scope of this study, and the classification was made as on the Githup Repository[6]. Accordingly, these six titles are listed as 'Trojan Horse', 'Mirai', 'CoinMiner', 'Backdoor', 'Downloader', 'Malware', and 'Other'. The most preferred malicious file was the Mirai variants, followed by Trojan Horse and CoinMiner as seen in the Figure 3. Those that cannot be typed but are considered malicious files are listed under the Other heading.

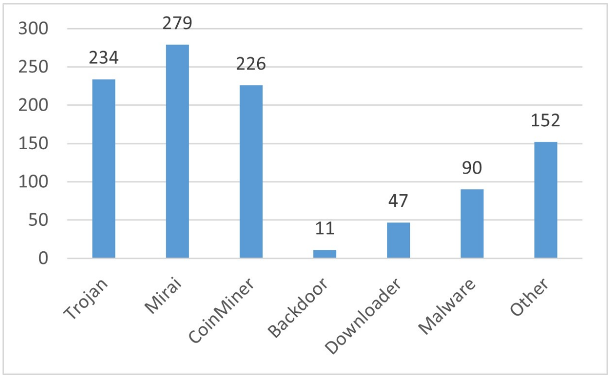


Fig. 3: Malware classification rates where suspicious files are tagged

The top 20 anti-virus applications on Virus Total used within this study's scope with the highest positive responses to 50 malicious files uploaded within a certain period are given in Figure II. Two of the 50 malicious files evaluated received no positive response from any anti-virus application. From this point of view, it can be interpreted that these two files are not harmful. According to these results, AVG is the anti-virus application with the highest number of positive responses, followed by Avast and DrWeb applications.

It has been observed that some anti-virus applications cannot classify uploaded files as 'harmful' or 'not harmful' or do not respond at all within a certain period. The 19 anti-virus applications that failed to classify or return an answer are given in Figure 4. In addition, 36 antivirus applications were able to detect less than ten malware.

VI. Discussion and Conclusion

We evaluated the files collected with Cowrie Honeypot, which we installed on Google Cloud through Virus Total. As part of the study, we developed a program script to evaluate 50 files obtained from the Cowrie system through Virus Total. We detected the files using the signature-based method in the study and compared the anti-virus tools.

[6]https://github.com/istec-iuc/cowrie-logs

TABLE II: Number of suspicious files that antivirus applications respond to as malicious

Anti-virus Applications	Positive Numbers
AVG	47
Avast	42
DrWeb	40
TrendMicro-HouseCall	40
TrendMicro	38
McAfee-GW-Edition	36
Fortinet	35
ESET-NOD32	34
Kaspersky	34
McAfee	34
GData	33
Symantec	30
BitDefender	28
Emsisoft	28
FireEye	28
Avira	27
Cynet	27
Jiangmin	26
Lionic	26
MicroWorld-eScan	26

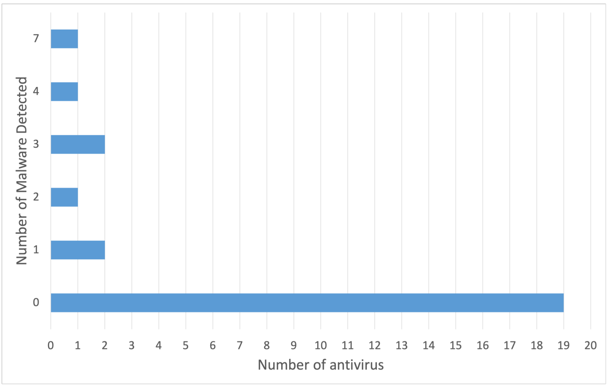


Fig. 4: Distribution of antiviruses with the most unsuccessful malware detection

Within the scope of the study, it aims to collect the actively used malicious software and classify these malicious software while simultaneously evaluating the success rate of anti-virus applications. According to the results, AVG is the anti-virus application with the highest success rate, followed by Avast and DrWeb applications. This study shows that AVG and Dr.Web's databases can recognize recent attacks by scanning files uploaded to a decoy system. Since the results were collected at the time of the attack, it is possible that antivirus applications update their databases later. The dataset shared in this study could be used in future research to compare which antivirus applications have updated themselves in the time since publication. The trap system, which was set up to catch current malware, was open for about 2 months and collected 50 malware samples. In a future study that is open for a longer period of time, more malware could be collected and the time it takes for antiviruses to update their signature databases against current attacks could be evaluated at different times of the year. On the other hand, there were opinions that the databases of 19 anti-virus applications, which could not be returned for 50 files, were not up-to-date or that the databases were too large to return a response within a certain period.

Signature-based detection is still a standard method used to detect malware. Of the anti-virus applications we tested, AVG achieved the most with 98 percent. After AVG, the highest was the AVAST application, with 87 percent success. However, when the top 10 anti-virus applications that returned the most positive responses were evaluated, it was seen that the success rate decreased to 70 percent. Even if the success rate of anti-virus applications is 99 percent, systems can be damaged if any malware infection is evaluated for corporate companies and end-users. Since it is generally not possible to run more than one anti-virus application on an operating system as is done within the scope of the study, applications with up-to-date and broad signature databases should be preferred. Generally, there are disadvantages such as anti-virus applications adversely affecting system performance by system requirements (processing power, ram, Internet, etc.) or not being used due to high license costs. For this reason, in our study, it was seen that in addition to multiple anti-virus application evaluations, malicious files increase the likelihood of not being detected because they cannot be tested on multiple anti-viruses. In this context, it can be considered a suitable solution for anti-virus applications to share databases with each other in order to make the systems more secure. Furthermore, virus Total runs multiple anti-virus applications, making it easy and advantageous to use multiple anti-viruses.

When examining the overall stats, the fact that approximately 76 percent of the requests respond negatively or do not respond at all makes the success of signature-based systems questionable, especially considering that there was no response from 19 antivirus applications. Of the 64 antivirus applications included in the evaluation, 19 did not receive a response, which raises several concerns regarding the performance of the antivirus applications themselves, the effectiveness of VirusTotal, the insufficient API response time, and the relevance of the up-to-date data set used in our study.

This study shows that while signature-based Anti-viruses with up-to-date databases are still successful, signature-based detection alone is not sufficient. It is understood that static and dynamic analysis methods are also necessary for system security. In our study, we developed a Python application that works on file signatures to quickly evaluate malware uploaded by attackers to the honeypot environment. In the future, we aim to collect more files with a trap system that will run for a longer period of time. We also aim to provide a more comprehensive and robust assessment of malware detection by incorporating behavior-based, heuristic and AI-assisted techniques into the analysis.

Acknowledgment

This study was carried out in the Internet of Things Security Test and Evaluation Center (ISTEC) of İstanbul University-Cerrahpasa and sponsored by the Istanbul Development Agency. Also, this publication was prepared within the scope of master's thesis.

References

[1] G. Pitolli, G. Laurenza, L. Aniello, L. Querzoni, and R. Baldoni, "Malfamaware: automatic family identification and malware classification through online clustering," *International Journal of information security*, vol. 20, pp. 371–386, 2021.
[2] M. Amal and P. Venkadesh, "Review of cyber attack detection: Honeypot system," *Webology*, vol. 19, no. 1, pp. 5497–5514, 2022.
[3] S. COOK, "Malware statistics in 2022: Frequency, impact, cost &amp; more," Feb 2022. [Online]. Available: https://www.comparitech.com/antivirus/malware-statistics-facts/
[4] S. S. Chakkaravarthy, D. Sangeetha, and V. Vaidehi, "A survey on malware analysis and mitigation techniques," *Computer Science Review*, vol. 32, pp. 1–23, 2019.
[5] N. Pachhala, S. Jothilakshmi, and B. P. Battula, "A comprehensive survey on identification of malware types and malware classification using machine learning techniques," in *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*. IEEE, 2021, pp. 1207–1214.
[6] C. Rohith and G. Kaur, "A comprehensive study on malware detection and prevention techniques used by anti-virus," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. IEEE, 2021, pp. 429–434.
[7] K. Oosthoek and C. Doerr, "Cyber threat intelligence: A product without a process?" *International Journal of Intelligence and CounterIntelligence*, vol. 34, no. 2, pp. 300–315, 2021.
[8] D. Aygör and E. Aktan, "The limitations of signature-based and dynamic analysis methods in detecting malwares: A case study," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 37, no. 1, pp. 305–315, 2022.
[9] U. Inayat, M. F. Zia, F. Ali, S. M. Ali, H. M. A. Khan, and W. Noor, "Comprehensive review of malware detection techniques," in *2021 International Conference on Innovative Computing (ICIC)*. IEEE, 2021, pp. 1–6.
[10] D. Laka, "Malware: Types, analysis and classification," *Analysis and Classification (January 14, 2022)*, 2022.
[11] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "Sok: cryptojacking malware," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 120–139.
[12] S. Talukder and Z. Talukder, "A survey on malware detection and analysis tools," *International Journal of Network Security & Its Applications (IJNSA) Vol*, vol. 12, 2020.
[13] S. A. Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," *Computers & Electrical Engineering*, vol. 92, p. 107143, 2021.
[14] S. Varlioglu, N. Elsayed, Z. ElSayed, and M. Ozer, "The dangerous combo: Fileless malware and cryptojacking," *SoutheastCon 2022*, pp. 125–132, 2022.
[15] T. Alsmadi and N. Alqudah, "A survey on malware detection techniques," in *2021 International Conference on Information Technology (ICIT)*. IEEE, 2021, pp. 371–376.
[16] A. Chavan, K. Kerakalamatti, and S. Srivastva, "Implementation of portable antivirus system using signature-based detection and heuristic analysis," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2021, pp. 1481–1486.
[17] M. Botacin, M. Z. Alves, D. Oliveira, and A. Grégio, "Heaven: A hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection," *Expert Systems with Applications*, vol. 201, p. 117083, 2022.

[18] M. J. H. Faruk, H. Shahriar, M. Valero, F. L. Barsha, S. Sobhan, M. A. Khan, M. Whitman, A. Cuzzocrea, D. Lo, A. Rahman *et al.*, "Malware detection and prevention using artificial intelligence techniques," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5369–5377.

[19] S. M. de Lima, H. K. d. L. Silva, J. H. d. S. Luz, H. J. d. N. Lima, S. L. d. P. Silva, A. de Andrade, and A. M. da Silva, "Artificial intelligence-based antivirus in order to detect malware preventively," *Progress in Artificial Intelligence*, vol. 10, no. 1, pp. 1–22, 2021.

[20] S. Rani, K. Tripathi, Y. Arora, and A. Kumar, "Analysis of anomaly detection of malware using knn," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, vol. 2. IEEE, 2022, pp. 774–779.

[21] A. Katkar, S. Shukla, D. Shaikh, and P. Dange, "Malware intrusion detection for system security," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*. IEEE, 2021, pp. 1–5.

[22] T. A. Assegie, "An optimized knn model for signature-based malware detection," *Tsehay Admassu Assegie." An Optimized KNN Model for Signature-Based Malware Detection". International Journal of Computer Engineering In Research Trends (IJCERT), ISSN*, pp. 2349–7084, 2021.

[23] M. Zyout, R. Shatnawi, and H. Najadat, "Malware classification approaches utilizing binary and text encoding of permissions," *International Journal of Information Security*, pp. 1–26, 2023.

[24] V. Sethia and A. Jeyasekar, "Malware capturing and analysis using dionaea honeypot," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–4.

[25] I. M. M. Matin and B. Rahardjo, "A framework for collecting and analysis pe malware using modern honey network (mhn)," in *2020 8th International Conference on Cyber and IT Service Management (CITSM)*. IEEE, 2020, pp. 1–5.

[26] A. Kyriakou and N. Sklavos, "Container-based honeypot deployment for the analysis of malicious activity," in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–4.

[27] C. Moore, "Detecting ransomware with honeypot techniques," in *2016 Cybersecurity and Cyberforensics Conference (CCC)*. IEEE, 2016, pp. 77–81.

[28] B. Wang, Y. Dou, Y. Sang, Y. Zhang, and J. Huang, "Iotcmal: Towards a hybrid iot honeypot for capturing and analyzing malware," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[29] J. Aycock, *Computer viruses and malware*. Springer Science & Business Media, 2006, vol. 22.

[30] R. Ball, "Computer viruses, computer worms, and the self-replication of programs," in *Viruses in all Dimensions: How an Information Code Controls Viruses, Software and Microorganisms*. Springer, 2023, pp. 73–85.

[31] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of malware threats and techniques: a review," *International Journal of Communication Networks and Information Security*, vol. 12, no. 3, pp. 326–337, 2020.

[32] "CNSSI 4009: Committee on national security systems (cnss) glossary," Committee on National Security Systems (CNSS), 2015, accessed: 2024-10-28. [Online]. Available: https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf

[33] J. Aycock, *Spyware and adware*. Springer Science & Business Media, 2010, vol. 50.

[34] I. Kuzminykh and M. Yevdokymenko, "Analysis of security of rootkit detection methods," in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*. IEEE, 2019, pp. 196–199.

[35] N. A. Mims, "Chapter 14 - the botnet problem," in *Computer and Information Security Handbook (Fourth Edition)*, J. R. Vacca, Ed. Morgan Kaufmann, 2025, pp. 261–272.

[36] M. Swanson and B. Guttman, "NIST SP 800-12 Rev. 1: An Introduction to Information Security," National Institute of Standards and Technology (NIST), Tech. Rep. 800-12 Rev. 1, 2017, accessed: 2024-10-28. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/12/r1/final

[37] A. Warikoo, "Perspective chapter: Ransomware," in *Malware-Detection and Defense*. IntechOpen, 2023.

[38] E. Salimi and N. Arastouie, "Backdoor detection system using artificial neural network and genetic algorithm," in *2011 International Conference on Computational and Information Sciences*, 2011, pp. 817–820.

[39] H. W. Kim, "A study on countermeasures by detecting trojan-type downloader/dropper malicious code," *International Journal of Advanced Culture Technology*, vol. 9, no. 4, pp. 288–294, 2021.

[40] A. Damodaran, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 1–12, 2017.

[41] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[42] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*, 2013, pp. 113–120.

[43] Y. K. B. M. Yunus and S. B. Ngah, "Review of hybrid analysis technique for malware detection," *IOP Conference Series: Materials Science and Engineering*, vol. 769, no. 1, p. 012075, feb 2020. [Online]. Available: https://doi.org/10.1088/1757-899x/769/1/012075

[44] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "An effective memory analysis for malware detection and classification," *Comput., Mater. Continua*, vol. 67, no. 2, pp. 2301–2320, 2021.

[45] K. Monnappa, *Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware*. Packt Publishing Ltd, 2018.

[46] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—a state of the art survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–48, 2019.

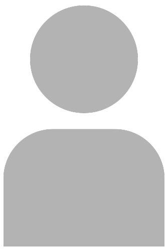
[47] Y. K. B. M. Yunus and S. B. Ngah, "Review of hybrid analysis technique for malware detection," in *IOP Conference Series: Materials Science and Engineering*. IOP Publishing, 2020, p. 012075.

[48] R. Sihwail, K. Omar, and K. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *Int. J. Adv. Sci. Eng. Inf. Technol*, vol. 8, no. 4-2, pp. 1662–1671, 2018.

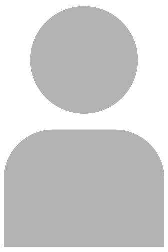
[49] M. Başer, E. Y. Güven, and M. A. Aydın, "Ssh and telnet protocols attack analysis using honeypot technique:* analysis of ssh and telnet honeypot," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2021, pp. 806–811.

[50] R. Masri and M. Aldwairi, "Automated malicious advertisement detection using virustotal, urlvoid, and trendmicro," in *2017 8th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2017, pp. 336–341.

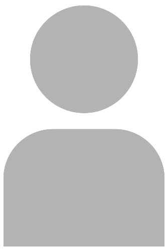
[51] A. Salem, S. Banescu, and A. Pretschner, "Maat: Automatically analyzing virustotal for accurate labeling and effective malware detection," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 4, pp. 1–35, 2021.



Melike BAŞER Melike Başer, completed her undergraduate education at Istanbul University-Cerrahpaşa with her graduation project titled ”SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot” in 2021. In 2022, she started working as a Research Assistant at Istanbul University-Cerrahpaşa Computer Engineering Department. In 2023, she completed his master’s degree at Istanbul University-Cerrahpaşa with his thesis titled ”Analysis of malicious files gathered on honeypot system”. She took part in 2 academic projects. In 2023, he started her PhD in Computer Engineering at Istanbul Technical University and continues her education.



Ebu Yusuf GÜVEN received his bachelor’s degree in Computer Engineering from Istanbul University in Turkey. He completed his master’s degree at Fatih Sultan Mehmet University with his thesis titled ”Cyber Attack Detection and Prevention Methods for Edge Computing”. In 2023, he completed his doctorate education at Istanbul University-Cerrahpaşa. His thesis is titled ”Development of a New Screening Model for Cyber Threat Intelligence”. He is currently working as an assistant professor at Istanbul University-Cerrahpaşa and a researcher at IoT Security Test and Evaluation Center (ISTEC). He has a keen interest in cyber security and related fields.



Muhammed Ali AYDIN obtained his B.S. degree in computer engineering from Istanbul University in Istanbul, Turkey in 2001. He completed his MSc degree in computer engineering from Istanbul Technical University, Istanbul, Turkey in 2005. He received his Ph.D. degree in computer engineering from Istanbul University, Istanbul, Turkey in 2009. He was a Postdoctoral Research Associate with the Department of RST, Telecom SudParis, Paris, France, from 2010 to 2011. He has been working as an Assistant Professor in Istanbul University-Cerrahpasa Department of Computer Engineering since 2009. He is the Vice Dean of Engineering Faculty and Head of Cyber Security Department since 2016. His research interests include optical networks, network security, information security and cryptography.

APPENDIX

TABLE III: Malware Classification of Suspicious Files - Trojan

Trojan
Trojan.CYGU-5
Trojan.GenericKD.37014839
Trojan.GenericKD.46282322 (B)
Linux.Trojan.Agent.WM6SLY
Trojan.GenericKD.36919466 (B)
Trojan.Generic.D233049B
Trojan:Linux/Multiverze
Trojan.Gen.NPE
E32/Trojan.OJUR-2
TROJ_GEN.R049C0PF82
Trojan/Generic.ASSuf.3CFBA
Trojan.Generic-Bash.Save.81cad571
Linux.Trojan.Agent.BYCO3E
Trojan.Linux.Agent
Trojan.Generic.D2C47968
Trojan.Generic-Bash.Save.b4766376
TrojanDownloader.Linux.w
Trojan-Downloader.Linux.Sh
Trojan.Generic.D2C23652
Linux.Trojan.Agent.U1BI7F
E32/Trojan.ADHC-11
Trojan.BDNM-12
Trojan.Malware.121218.susgen
Trojan.SuspectCRC
Unix.Trojan.DarkNexus-7679166-0
Trojan.Shell.Agent
Trojan.GenericKD.37014839 (B)
Trojan.ElfArm32.Dwn.ixvyeu
Trojan.Generic
Trojan.GenericKD.36890669 (B)
TrojanDownloader.Linux.yr
Trojan.GenericKD.46111158 (B)
Trojan.Gafgyt.Linux.28527
Trojan.GenericKD.46354653
Trojan.GenericKD.37044452 (B)
Elf.Trojan.A1686007
Trojan.GenericKD.46111158
Linux.Trojan.Agent.956L7O
Trojan.Generic.D2C3EBF0
Trojan.U.Downloader.oa
Trojan.Generic-Script.Save.ba695
Trojan.Shell.Agent.a!c
E32/Trojan.KSLF-8
Trojan.GenericKD.46373334 (B)
ELF.Trojan.42987.GC
TrojanDownloader.Linux.xc
Linux.DownLoader.598
E32/Trojan.BIKP-4
Archive.Trojan.Agent.237VFG
Trojan.GenericKD.46300839
Win32.Trojan-downloader.Agent.Ljke
Trojan.GenericKD.36946419
Trojan:Script/Woreflint.A!cl
Trojan.GenericKD.36897947 (B)
TROJ_GEN.R002C0DDR21
Trojan.Generic.D233C1F3
TROJ_FRS.VSNW17D21
Trojan.Generic.D233C1F3

TABLE IV: Malware Classification of Suspicious Files - Other

Other
GenericRXPJ-JL!B7740B0E0D53
GenericRXPL-SI!3B6C0968416D
Script.Application.Agent.ESB7CE
multiple detections
LINUX.Agent
Suspicious.Linux.Save.a
a variant of Generik.JHEILIY
PUA.Generic
Linux/DDoS-BIB
a variant of Linux/Gafgyt.AXI
GenericRXHV-CO!494F5DC3C97D
GenericRXID-MY!CEAE104121FB
Script.Application.Agent.BYSNFG
ex_virus.script.bash.000002
LINUX/Agent.SH.CDWC
GenericRXIA-AZ!3DC11C062A9F
LINUX/Gafgyt.quxje
BV:Agent-BHX [Drp]
GenericRXLU-EV!62CF685B889F
LINUX/Dldr.Agent.chw
GenericRXIA-VB!2CA7C03A8DCB
Linux.Lightaidra
PUP-XNS-AN
GenericRXHZ-TQ!F42C2BC71DC8
GenericRXIA-WK!D0B5FB9EA23F
a variant of Generik.GVFDFVN
Linux/DDoS-CIA
ELF:CVE-2017-17215-A [Expl]
PossibleThreat
Linux/Gafgyt.DF!tr.bdr
GenericRXIB-LH!FC4CD46E3F7C
HackTool.XMRMiner!1.C0AC (CLASSIC)
GenericRXIB-BH!F3630D8BF607
RDN/Generic.dx
GenericRXIB-FF!4CDE0578155B
RiskTool.Linux.clc
Static AI - Malicious ELF

TABLE V: Malware Classification of Suspicious Files - Mirai

Mirai
avariantofLinux/TrojanDownloader.Mirai.A
LINUX/Dldr.Mirai.nmpob
Trojan.Linux.Mirai.GED
Generic.Bash.MiraiA.EB5DB9B6(B)
Generic.Bash.MiraiA.EB5DB9B6
LINUX/Dldr.Mirai.aymds
LINUX/Dldr.Mirai.gours
HEUR:Trojan-Downloader.Linux.Mirai.d
ELF/Mirai.A!tr.dldr
Backdoor.Linux.MIRAIA.USNELFD21
Gen:NN.Mirai.34058
Linux/Mirai.Gen35
Generic.Bash.MiraiB.F5F3FD03(B)
LINUX/Mirai.bykao
Backdoor.Mirai/Linux!1.BAF6(CLASSIC)
Possible_MIRAIDLOD.SMLBAT5
Trojan.Linux.Mirai.K!c
Backdoor.Mirai/Linux!1.B311(CLASSIC)
UDS:Backdoor.Linux.Mirai
Linux/Mirai.D!tr.dldr
Linux.Trojan-downloader.Mirai.Wqmn
Backdoor.Linux.MiraiDownload.ve
Linux.Mirai.B!tr.bdr
ELF:MiraiDownloader-KG[Trj]
Linux.Mirai.2924
HEUR:Trojan-Downloader.Linux.Mirai.e
ELF/Mirai.D!tr
ELF:MiraiDownloader-JD[Trj]
Lnx/Mirai-FEBO!62CF685B889F
LINUX/Mirai.apkfl
Linux.Mirai.4511

TABLE VI: Malware Classification of Suspicious Files - Backdoor and Malware

Backdoor	Malware
Backdoor.Linux.rec	Malware@#1hds26rs9a7iq
Backdoor.Linux.ZYX.USDSEED21	Malware.Generic-Script.Save.ba303
Backdoor.Linux.bskw	Malware@#t8qixr26txeu
Backdoor.Linux.Agent	Mal/ShellDl-A
HEUR:Backdoor.Linux.Gafgyt.df	Malware@#z42j21fz2ven
Backdoor/Text.CryptoBot	Malware@#11tskun56nig
RDN/Generic BackDoor	Malware.Generic-Script.Save.ba104
	Other:Malware-gen [Trj]
	Malware.ELF-Script.Save.b606d819
	Malware@#3noqscxynq9q3

TABLE VII: Malware Classification of Suspicious Files - CoinMiner and Downloader

CoinMiner	Downloader
Multios.Coinminer.Miner-6781728-2	Shell/ElfDownloader.S3
E64/CoinMiner.B.gen!Camelot	Linux.DownLoader.535
Downloader/Shell.ElfMiner.S1165	Linux.DownLoader.533
Application.Linux.Miner.LL	BV:Downloader-JS [Drp]
Trojan.U.CoinMiner.oa	TrojanDownloader.Linux.eh
Linux/CoinMiner.Gen2	Linux/Downloader.1288
ELF/BitCoinMiner.HF!tr	RDN/Generic Downloader.x
not-a-virus:HEUR:RiskTool.Linux.BitCoinMiner.b	Linux/Downloader.w
LINUX/BitCoinMiner.fmbfm	Linux.DownLoader.532
Linux.Risk.Bitcoinminer.Dwtp	Linux.DownLoader.459