Malware Analysis Through High-level Behavior

Xiyue Deng and Jelena Mirkovic
{xiyueden,mirkovic}@isi.edu
Information Sciences Institute

Abstract

Malware is becoming more and more stealthy to evade detection and analysis. Stealth techniques often involve code transformation, ranging from equivalent code substitution and junk code injection, to continuously transforming code using a polymorphic or a metamorphic engine. Evasion techniques have a great impact on signature-based malware detection, making it very costly and often unsuccessful.

We propose to study a malware's network behavior during its execution. While malware may transform its code to evade analysis, we contend that its behavior must mostly remain the same to achieve the malware's ultimate purpose, such as sending spam, scanning for vulnerable hosts, etc. While live malware analysis is hard, we leverage our Fantasm platform on the DeterLab testbed to perform it safely and effectively. Based on observed network traffic we propose a behavior classification approach, which can help us interpret the malware's actions and its ultimate purpose at a high level. We then apply our approach to 999 diverse samples from the Georgia Tech Apiary project to understand current trends in malware behaviors.

1 Introduction

Contemporary state of Internet security is grave, thanks to proliferation of malware. Malware is an increasing threat, not only because of its increased capability and sophistication, but also because of its stealth, aimed at avoiding detection and analysis.

Because anti-virus suites usually apply signature-based malware detection, malware designers have invested enormous effort to change their binaries to avoid detection. From simple instruction transformation, such techniques have evolved through junk code injection, code obfuscation, to polymorphic and metamorphic engines that can transform malware code into millions of variants [17]. Such obfuscation techniques have greatly undermined traditional signature-based malware detection methods. We simply cannot keep pace with proliferating malware variants. Notable attempts have been made by researchers to improve signature analysis by running AV suites on cloud [22], detecting encryption decoders [15], detecting runtime signatures for polymorphic malwares [24], etc. However, as new obfuscation techniques emerge, signature analysis alone cannot save us.

Besides static analyses, researchers have used dynamic analysis to overcome code obfuscation. These include tracking disk changes, analyzing dynamic call graphs, as well as monitoring malware execution using debuggers and virtual machines. However, modern malware is often equipped with anti-debugging and anti-virtualization capabilities [18, 19], making its analysis in a controlled environment difficult. Yet, a controlled environment is needed to observe a malware's actions on the compromised host, and record sufficient details for analysis.

To complement contemporary static and dynamic *code* analysis, we propose to study malware behavior by observing and interpreting its network activity. Based on our prior research [6], we believe that network access is essential for malware to achieve its purpose. We also believe that it would be difficult for malware to significantly alter its network behavior and still achieve its purpose. Studying network behavior thus may offer an opportunity to both understand what malware is trying to do in an analysis environment, and to detect malware on compromised hosts. In this paper we pursue this first direction, by investigating how we can infer the malware's ultimate purpose by analyzing its network activity.

Live malware analysis, with unrestricted network access, is risky, because malware may inflict damage to other Internet hosts during analysis, and we would become unwitting accomplices. We leverage our Fantasm system for safe and productive live malware analysis [6]

to minimize this risk.

Just having network traffic records of a malware run is not enough to understand a malware's purpose, because such data is very rich and very unstructured. To overcome this challenge, we first extract a set of features from network traffic records, and then devise patterns that help us group these features into higher-level behaviors, such as file download, e-mail sending, scanning, etc. Next we use a set of these high-level behaviors to label malware by its purpose. We note that malware could have more than one purpose. For example, it could both scan for vulnerable hosts and send unsolicited emails. Thus a malware sample could have more than one label.

We perform our analysis on 999 malware samples, randomly selected from the Georgia Tech Apiary project [1]. We observe that the most common activities are scanning, propagating, and downloading. Interestingly, of all samples that show more than one behavior, scanning and propagating show up most frequently together, which suggests these activities usually work in combination and complement each other.

Our paper is structured as follows. We discuss background of this work and contemporary research in Section 2. In Section 3 we explain our methodology. We define our classification criteria in Section 4; We describe experiment set up and evaluation in Section 5. We discuss the use of the results and possible future work in Section 6 and conclude in Section 7.

2 Background and Related Work

In this section we discuss contemporary malware analysis methods and the rationale behind our research methods, and also survey related work.

2.1 Static Analysis

Various static analysis methods like CTPL [11], Generic Virus Scanner [5], etc. have been used to analyze binary code of malware without running it, and identify portions that could be used for signature generation. These portions of code exhibit malicious activity, and are unlikely to present in benign binaries. The identified portion is then synthesized to become the signature of this kind of malware. Once the signature is obtained, researchers can build anti-virus tools to find the same kind of malware in future by scanning all suspicious binaries.

Signature-based malware detection has been the most widely used method and has been quite successful. However, malware designers have also been working on countermeasures over the years to undermine such techniques. From junk code generation, malware encryption and oligomorphism, to polymorphic and metamorphic malware [17], such techniques have evolved significantly against signature-based detection techniques. On the other hand, researchers have also been improving signature detection methods, such as detecting decryption routines, runtime signature detection, etc. Overall, this is a losing race for the researchers, as it will always be cheaper to generate new malware variants, than to analyze them. We can become faster, but we can never win. Our proposed direction shows promise because malware cannot change its network behavior with as much freedom as changing its code. We thus believe that we will be able to analyze and detect those malware samples that static analysis misses.

2.2 Dynamic Analysis on Host

Instead of looking for a binary signature to detect malware presence, dynamic analysis builds a signature of a malware's interaction with its host. This signature may include memory access and file access patterns, as well as system call patterns. Those patterns that are likely to be present in malware but not in benign binaries can be used to develop a behavioral signature for malware detection [7].

Dynamic analysis complements static analysis, and is able to overcome malware code obfuscation. Willems et al. [23] proposed CWSandbox that combines static and dynamic techniques for analyzing malware on a contained host. Guarnieri et al. [8]'s Cuckoo sandbox follows similar concepts as CWSandbox and additionally provides a network packet sink. Both works utilize virtual machines to isolate study environment and the host. However, stealthy malware has another set of techniques to evade dynamic analysis – it detects debuggers and virtual machines, which are often used to speed up and facilitate dynamic analysis, and modifies its behavior. This leads to incorrect or unusable signatures of stealthy malware. Chen et al. [4] found that 39.9% and 2.7% of 6,222 malware samples exhibit anti-debugging and anti-virtualization behaviors respectively. In 2012, Branco et al. [3] analyzed 4 million samples and observed that 81.4% of them exhibited anti-virtualization behavior and 43.21% exhibited anti-debugging behavior.

Our work helps analyze malware that would otherwise be missed by static and dynamic analysis, because it relies on observations of malware's network behavior. Malware can be run on bare metal machines thus evading detection of the analysis environment.

2.3 Dynamic Analysis of Network Behavior

There are a few efforts on analyzing the semantic of malware network behavior. Sandnet [16] provided a detailed, statistical analysis of malware network traffic.

The authors surveyed protocol popularity employed by malware. However, they did not go further to understand the semantic of the network communication, while we do.

Morales et al. [13] studied seven network activities, selected using heuristics. These activities could be used to identify malicious activities. Morales et al. also report on prevalence of those behaviors in contemporary malware. Their chosen activities are not as useful to understand a malware sample's purpose or type, and are very limited. We complement this work by extending the set of activities being observed.

Nari et al. [14] proposed an automated malware classification system also focusing on malware network behavior, which generates protocol flow dependency graph based on the IP address being contacted by malware. Our work improves this effort by systematizing detection of different malware behaviors using different network behavior patterns, and different network traffic contents.

3 Capturing Malware Network Behavior

With prevalence of high-speed Internet access, malware has begun to rely more and more on network to achieve its ultimate purpose [9, 20]. Malware often downloads binaries needed for its functionality from the Internet, or connects into command and control channel to receive instructions on its next activity [10]. Advanced persistent threats [21] and keyloggers collect sensitive information on users' computers, but need network access to transfer it to the attacker. DDoS attack tools, scanners, spam and phishing malware require network access to send malicious traffic to their targets.

We propose to study malware's network activities because they have become essential for modern malware. The first step in this study includes capturing malware's network traffic in an environment, which is transparent to malware but also minimizes risk to Internet hosts.

3.1 Analysis Environment: Fantasm

In our prior work, we have developed a semi-contained network experiment environment, called Fantasm [6]. Fantasm runs on a testbed with full Internet access, and carefully constrains this access to achieve productive malware analysis, and minimize risk to outside hosts. In our analysis, we run malware on a bare metal Windows host, without any virtualization or debugger. We capture and analyze all network traffic between this machine and the outside using a separate Linux host, sitting in between the Windows host and the Internet. Both hosts are controlled by our Fantasm framework.

Fantasm makes decisions on which communications to *impersonate*, i.e., intercept and answer itself, which

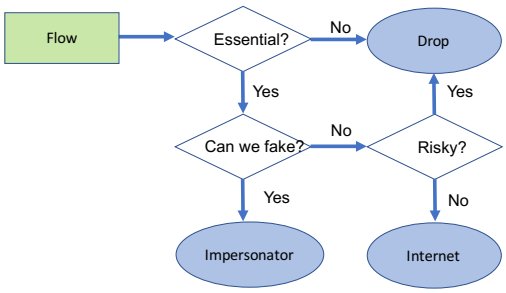

Figure 1: Flow handling: how we decide if an outgoing flow will be let out, impersonated or dropped.

to *forward* and which to *drop*. This decision is made by taking into account each outgoing flow separately, making an initial decision, and revising it later if subsequent observations require this. We define a flow as a unique combination of destination IP address, destination port and protocol. Each flow is initially regarded as *non-essential*, and it is dropped. If this leads to the abortion of malware activity, we stop our analysis, restart it, and regard that specific flow as *essential*. Fantasm then considers if it can fake replies to this outgoing connection in a way that would be indistinguishable from the actual replies, should the flow be allowed into the Internet. If Fantasm has an *impersonator* for the given destination and the given service, it will intercept the communication and fake the response. Otherwise, it will evaluate if the outgoing flow may be risky to let out. If so, the flow will be dropped. Otherwise, it will be let out into the Internet. Table 1 illustrates the criteria we use to determine if a flow is risky or not, and if it can be impersonated. Note that even flows considered not risky and let out are still subjected to further monitoring and may be aborted if they misbehave.

Flows, which are let out, could become part of scanning or DDoS. To minimize risk of unwitting participation in attacks, we actively monitor for these activities and enforce limits on the number of *suspicious* flows that a sample can initiate. We define a suspicious flow as a flow, which receives no replies from the Internet. Many scan and DDoS flows will be classified as suspicious. If the analyzed malware sample exceeds its allowance of suspicious flows (10 in our current implementation), we abort its analysis.

3.1.1 Impersonators

To handle simple outgoing requests with predictable replies, we built dummy services that can produce these replies within our analysis environments, called *impersonators*. Several protocols used by popular services are good candidates to be handled by impersonators, such as ICMP, SMTP, and FTP. ICMP_ECHO messages have a

Service or Protocol	Label
DNS, HTTP, HTTPS	Not risky
FTP, SMTP, ICMP_ECHO	Risky, can be impersonated
Other services or protocols	Risky, cannot be impersonated

Table 1: Flow policies in Fantasm

predictable reply, which can be either positive or negative. Our impersonator provides a positive reply to each ICMP_ECHO request. SMTP actions by malware are usually used for spamming purpose, and require us to only fake a successful receipt. We do this by setting up an Email server, which replies with a "250 OK" message to any request. Our FTP service impersonator is a customized, permissive FTP service, which accepts any user name and password combination.

In addition to impersonating servers for these three types of traffic, we set up our Fantasm host to act as DNS caching proxy for all DNS requests by malware. This enables us to observe malware's DNS traffic and to cache replies, thus again minimizing outgoing requests.

4 High-level Behaviors

To analyze captured network traces, and reason about malware behavior, we need to impose some structure on trace data. We do this by identifying certain network behavior patterns, which we believe will be commonly used for a given malware purpose. We call these patterns *high-level behaviors*. We seek to identify behaviors, which are common for the following malware purposes:

- **Downloading**. This type of malware activity is prevalent and usually occurs immediately upon infection, since many malware samples are just empty downloaders to defeat static and dynamic analysis [2].
- **Reporting**. This type of activity usually suggests that malware samples are submitting information gathered from the victim to their command & control server, as well as requesting more information from it.
- **Scanning**. This is usually used to collect vulnerable host identities, to use as potential victims in future attacks.
- **Spamming**. Such malware samples send spam emails that usually contain phishing materials like malicious URLs or attachments.
- **C&C communication**. Malware samples maintain a connection with certain public services such as IRC to receive C&C commands.
- **Propagating**. This type of malware takes advantage of file transfer protocols (e.g. ftp, samba, NFS, etc.) to propagate binaries to other hosts.

Table 2 summaries high-level behaviors, which we use to infer the above mentioned malware purposes:

4.1 High-level Behavior Signature

Fantasm extracts many features from network traces. These features could be used by machine learning to learn how to classify malware behaviors into high-level categories, but this would require manual labeling of many samples. Instead, we can use a simpler set of parameters and behavior patterns to achieve identification of high-level behaviors in a deterministic fashion. We utilize information including protocol, port, packet length, as well as some application level headers. We also observe the sequences of network behaviors to identify patterns that denote a specific high-level behavior. For example, if malware attempts to connect to an external host on port 25, 465 or 587 and send email we will infer that this is part of the "spamming" category. A detailed description of network patterns and parameters we use is shown in Table 2.

5 Experiment and Evaluation

We now use the patterns identified in the previous section to study prevalence of our selected high-level behaviors in contemporary malware.

5.1 Experiment Setup

Our experiment environment is built using the Fantasm platform [6], which allows for safe and productive live malware experimentation. Fantasm is built on the DeterLab testbed [12], which is a public testbed for cybersecurity research and education. DeterLab allows users to request a number of physical nodes, connect them into custom network topologies and install custom OS and applications on them. Users are granted root access to the machines in their experiments. Fantasm utilizes this infrastructure to construct a LAN environment with a node running Windows to host the malware, and another node running Ubuntu Linux acting as the gateway.

Action	Network Activity Features	Detection parameters
Downloading	Connection to remote server Receiving payload for further action	HTTP GET request for a specific URL Receive payload (binary or text)
Reporting	Connection to remote server Submit data to remote server	HTTP POST to remote server with parameters or payload
Scanning	Detecting availability of other hosts Detecting open ports of hosts	ICMP ECHO packets to other hosts TCP SYN or UDP packets to ports
Spamming	Connection to public mail server Sending spam email	TCP connection to port 25, 465, or 587 Text email with URL or attachments
C&C Communication	HTTP, IRC connection to remote server Receive instruction	IRC protocol / ports
Propagating	Try to transfer data over to other hosts	FTP, Samba, NFS protocol / ports

Table 2: High-level behavior list.

Fantasm provides necessary services for impersonators, and monitors the network activities by capturing all network packets using *tcpdump*. One round of analysis for a given malware sample consists of the following steps:

- Enable service network monitoring on Linux gateway
- Reload operating system on Windows node, and set up necessary network configurations
- Deploy and start the malware binary and continue running it for a given time period (we used 5 minutes)
- Kill the malware process and save the captured network trace.

This setting has the advantage that it is immune to current analysis evasion attempts by malware, because it does not use a debugger or a virtual machine. By reloading OS for each run, it ensure that each sample is analyzed in an environment free from any artifacts from the previous analysis rounds.

The malware samples we used in the study were provided by the Georgia Tech Apiary project [1], which is a repository with a daily collection of malware samples. We randomly selected 999 malware captured in Mar. 2016 for this research. Next we submitted each sample to VirusTotal [22] to determine the type of malware, and ensure that the sample is recognized as malicious. We then analyzed each selected sample in our experiment environment, using the method described above. After the analysis, we have saved traces with all captured communications to and from the Windows node.

5.2 Gathering High-Level Behavior

Our analysis program extracted all useful parameters to identify high-level behavior as described in Table 2. The program first generates raw results for all parameters such as protocol, destination ports, HTTP request type, remote hosts, as well as session sequence patterns. We then automatically identify patterns listed in Table 2, to detect high-level behaviors. Our results are shown in Table 3 and Table 4.

Behavior	Percentage
Downloading	10.9%
Reporting	5.6%
Scanning	**28.5%**
Spamming	2.2%
C&C communication	0.2%
Propagating	11.5%
Not Detected	57%

Table 3: Percentage of our selected 999 malware samples, which exhibit the given high-level behavior.

Count of behaviors	Ratio
1	67.6%
2	28.2%
3	3.4%
4	0.7%
5 or more	0%

Table 4: Percentage of malware having the given number of behaviors, among all malware, which exhibited some network activity.

Table 3 shows prevalence of different behaviors in our malware samples. Around 57% of the samples do not show any network activity. Possible reasons of inactivity include shortness of the observation period, malware waiting for an external trigger, or outdated (dormant) malware. We plan to establish the exact causes of inactivity in our future work.

Out of malware samples, which exhibit some network

Behavior Combinations	Ratio
Scanning+Propagating	43.1%
Downloading+Propagating	16.5%
Scanning+Spamming	12.2%
Downloading+Scanning	7.9%
Downloading+Scanning+Propagating	5.7%
Downloading+Uploading	4.3%
Uploading+Scanning	2.2%
Downloading+Uploading+Scanning+Propagating	2.2%
Downloading+Uploading+Propagating	1.4%
Scanning+Spamming+Propagating	1.4%
Uploading+Scanning+Propagating	1.4%
Uploading+Propagating	0.7%
Downloading+Uploading+Scanning	0.7%

Table 5: Percentage of malware exhibiting given combinations of behaviors.

activity, 28.5% engage in scanning scanning, 11.5% in propagating itself further and 10.9% in downloading new binaries. This suggests that most malware focuses on finding vulnerable hosts, downloading the actual malicious code and propagating further. Reporting shows up in 5.6% of the samples, while spamming shows up in 2.2% of the samples. Other types of behaviors, such as C&C communication, are rarely seen, which suggests we may need more samples or longer observations, to study these behaviors.

We were further interested in combinations of behaviors, which show up together. Table 4 shows the percentage of samples, which show one or more behavior. More than 30% of the samples, which exhibit some network activity, engage in more than one behavior, and around 4% of the samples engage in more than two behaviors. Table 5 shows the most common behavior combinations. Scanning and propagating show up most frequently together (43.1%), which is not surprising. This combination suggests that most propagating malware samples are also equipped with scanning capability to find potential victims. The next common combination is downloading with propagating, which suggests this kind of malware samples usually download new payload, and then propagate further. Also frequent is scanning with spamming, suggesting spam malware samples also scan for potential victims, before sending spam.

6 Discussion and Future Work

Currently we have defined patterns, which we can be used to detect six types of high-level behaviors. We believe there are more high-level behaviors that are useful for understanding malware, and we will seek to define them (and the associated patterns) in our future work.

Our experiment results also suggest that behavior combinations are common. We believe this finding can be useful for the following future directions in malware analysis.

Malware detection. Most low-level malware behaviors also occur in benign software, which makes them unreliable for malware detection purposes. However, combinations of these low-level behaviors may be unique to malware, and could be useful for detection. For example, a software that scans other hosts and then copies data over is unlikely to be benign, although each of these actions separately could be undertaken by benign software (e.g., probing several servers could look like scanning, if servers are unresponsive, and data can be uploaded to a cloud for legitimate reasons) As we extend our malware analysis to more samples, we expect to find more behavior combinations, which can be used for malware detection.

Polymorphic malware. Modern malware may take advantage of a polymorphic engine to encode itself and evade signature-based detection. Through network behavior analysis and behavior combination patterns, we may detect similar malicious behaviors exhibited by different binary samples. These samples could be polymorphic versions of the same code. Thus our work can be useful to detect polymorphic malware.

Malware genealogy. With longer analysis on more samples, we hope to build a more fine-grained representation of high-level malware behaviors. Such information can then be used identify malware samples, which have similar behaviors, and thus may be related. For example, if two samples send emails with the same attachments, use same or similar phishing URLs or contact the same C&C server, they are likely to be related. Similarly, if two samples exhibit similar spamming behavior (e.g., using same or similar content) but one has self-propagating behavior too, while the other does not,

these samples are likely to be related. Our high-level behaviors can be useful to study malware genealogy and to identify malware families. Similarly, when malware downloads a new binary, which then attempts to perform some malicious activity, these two samples work in collaboration for a common goal. We hope to leverage our current work to build information on such related malware in the future.

Malware Evading Network Analysis. With the existence of techniques to evade signature-based malware detection, we can naturally assume that malware designers will also look to evade our network-communication-based analysis. There are several strategies that malware could use:

(a) appear to be dormant and await a specific trigger

(b) interleave malicious activities with benign traffic

(c) attempt to detect analysis environment, e.g., by attempting to connect to an FTP server with non-existing username and password (this will succeed in our analysis but fail in real world)

We leave handling of these evasion techniques for our future work, but sketch some possible solutions here. For trigger-based malware, we could leverage existing work on trigger detection, such as using static analysis and symbolic execution [5] . For malware, which interleaves its malicious behavior with benign ones, we need to improve our pattern detection to consider subsets of activities, and not just sequences. For malware, which detects our impersonators, we plan to make our impersonators more sophisticated and harder to detect. There is however a trade-off here between impersonator sophistication and scalability. For example, our FTP impersonator could check for each username/password input, if this input can be used to access the actual FTP server in the real Internet. This information could be cached for the next attempt with the same input. Such approach would guarantee that our impersonator always provides correct reply to malware, but it would inundate us with testing and caching potentially many username/password combinations. This approach clearly does not scale. In our future work we will investigate where to draw the line between trying to improve sophistication and stealthiness of our impersonators, and scalability.

7 Conclusion

In this work we propose patterns of malware's network behaviors, which can be used to identify some high-level behaviors, such as scanning or spamming. We then study prevalence of these high-level behaviors in contemporary malware. We find that scanning, propagation and downloading are the most prevalent behaviors. We also find the some behavior combinations are quite popular, such as scanning and propagating, downloading and propagating, and scanning and spamming. We discuss how our identification of high-level behaviors could be useful for malware detection, identification of polymorphic malware and identification of malware samples, which may be versions of each other, or share some common code.

8 Acknowledgments

This project is the result of funding provided by the Science and Technology Directorate of the United States Department of Homeland Security under contract number HSHQDC-16-C-00024. The views and conclusions contained herein are those of the authors and should not be interpreted necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security or the US Government.

References

[1] Apiary. `http://apiary.gtri.gatech.edu/`. Accessed: 2018-05-09.

[2] M. Baggett. Effectiveness of antivirus in detecting metasploit payloads. *SANS Institute*, 2008.

[3] R. R. Branco, G. N. Barbosa, and P. D. Neto. Scientific but not academical overview of malware anti-debugging, anti-disassembly and anti-vm technologies. *Black Hat*, 2012.

[4] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 177–186. IEEE, 2008.

[5] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. Technical report, WISCONSIN UNIV-MADISON DEPT OF COMPUTER SCIENCES, 2006.

[6] X. Deng, H. Shi, and J. Mirkovic. Understanding malware's network behaviors using fantasm. *Proceedings of LASER 2017 Learning from Authoritative Security Experiment Results*, page 1, 2017.

[7] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis

techniques and tools. *ACM computing surveys (CSUR)*, 44(2):6, 2012.

[8] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser. The cuckoo sandbox. `https://cuckoosandbox.org/`, 2012.

[9] T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2009.

[10] K. T. D. Y. Huang, D. W. E. B. C. GrierD, T. J. Holt, C. Kruegel, D. McCoy, S. Savage, and G. Vigna. Framing dependencies introduced by underground commoditization. In *Workshop on the Economics of Information Security*, 2015.

[11] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith. Detecting malicious code by model checking. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 174–187. Springer, 2005.

[12] J. Mirkovic and T. Benzel. Deterlab testbed for cybersecurity research and education. *Journal of Computing Sciences in Colleges*, 28(4):163–163, 2013.

[13] J. A. Morales, A. Al-Bataineh, S. Xu, and R. Sandhu. Analyzing and exploiting network behaviors of malware. In *International Conference on Security and Privacy in Communication Systems*, pages 20–34. Springer, 2010.

[14] S. Nari and A. A. Ghorbani. Automated malware classification based on network behavior. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 642–647. IEEE, 2013.

[15] B. B. Rad, M. Masrom, and S. Ibrahim. Camouflage in malware: from encryption to metamorphism. *International Journal of Computer Science and Network Security*, 12(8):74–83, 2012.

[16] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. Van Steen, F. C. Freiling, and N. Pohlmann. Sandnet: Network traffic analysis of malicious software. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 78–88. ACM, 2011.

[17] A. Sharma and S. K. Sahay. Evolution and detection of polymorphic and metamorphic malwares: A survey. *arXiv preprint arXiv:1406.7061*, 2014.

[18] H. Shi, A. Alwabel, and J. Mirkovic. Cardinal pill testing of system virtual machines. In *USENIX Security Symposium*, pages 271–285, 2014.

[19] H. Shi and J. Mirkovic. Hiding debuggers from malware with apate. In *Proceedings of the Symposium on Applied Computing*, pages 1703–1710. ACM, 2017.

[20] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. *LEET*, 11:4–4, 2011.

[21] C. Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[22] VirusTotal. Virustotal-free online virus, malware and url scanner. `https://www.virustotal.com/en`, 2012.

[23] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, 5(2), 2007.

[24] I. You and K. Yim. Malware obfuscation techniques: A brief survey. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 297–300. IEEE, 2010.