International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7207-7214

Malware Detection and Evasion with Machine Learning Techniques: A Survey

Jhonattan J. Barriga A.[1] and Sang Guun Yoo [2]

Faculty of Systems Engineering, National Polytechnic School, Quito, Ecuador.

[1]ORCID ID: 0000-0001-7334-9113, Scopus Author ID: 57194788272
[2]ORCID ID: 0000-0003-1376-3843, Scopus Author ID: 36187649600 (Corresponding author)

Abstract

Malware has become a powerful and sophisticated tool used by malicious users to compromise and harm systems, and its evasion ability has improved considerably, getting to the point of becoming completely undetectable. On the other hand, machine learning has evolved tremendously in last years and it has become a standard in many IT solutions including the data processing field. Likewise, cryptography also has growth in popularity in providing confidentiality and integrity to important information. Even though those technologies are being widely used for trustable IT solutions, they also are used by malicious applications such as ransomware, which uses the cryptography as its infecting mechanism and the machine learning as its evasion technique. In this aspect, this paper makes a survey of existing researches regarding to malware detection and evasion by examining possible scenarios where malware could take advantage of machine learning and cryptography to improve its evasion techniques and infection impact.

Keywords: Malware, obfuscation, evasion, detection, machine learning malware

INTRODUCTION

Malware has turned into a powerful weapon for attackers that aim to take advantage of major security flaws. For this reason, malware production has increased considerably in the last three decades [1]. In the same way, there have been many efforts to stop malwares, and there have been developed different techniques based on signature, behavior, heuristic, sandboxing, and artificial intelligence approaches, among others. Using the aforementioned techniques, several works have proposed different solutions to stop malware [2]–[4]. But others have put their effort on improving the abilities of malicious software [5]–[9]. In this aspect, we can say that researching about malware becomes very important because it allows security analysts to understand the attacker's perspective in order to protect computers against them.

The increase of malware types has gained rebound in rewards of fields of infection and complexity of development. In fact, malware authors are more neat and pretend to gain the respect of their adversaries being quieter at the moment of infecting a computer, using elements such as social engineering or even spreading through the web [10].

In this paper, we briefly review malware detection techniques based on traditional approaches as well as novel ones that use artificial intelligence. Then, we focus on malware evasion techniques by analyzing approaches like obfuscation, polymorphism, GPU-assisted, among others. Finally, we discuss the use of Machine Learning techniques for evading anti-malware solutions based on previous works and ways to combine such techniques.

MALWARE DETECTION TECHNIQUES

An antivirus is a program that has the ability to scan several types of files on the disk, by comparing it with a known database. It is able to collect original file size and compare it within the time to validate that it has not grown [11]. It also can scan over the Master Boot Record (MBR), boot sectors, bad sectors, among others to determine if it has been infected. In regards of malware detection methods, three main categories have been recognized [1], [7], [8]: Signature based, Behavior based, and Heuristic based methods.

A. Signature Based Method

This technique searches sequences of bytes in order to identify a particular piece of malicious software. A signature is composed of a particular sequence of code or data. This signature is stored on a database which is used to compare to the scanned files. This is the most common method used to detect malware, since it produces a small error rate as described by [12]. However, there are some limitations. For example, it cannot detect new malwares since their signatures have not been generated yet. Additionally, internet connectivity is a must to download new signatures from the

server and keep the host protected [4]. Furthermore, since new malware appears every day, it is necessary to store large amounts of signatures, demanding considerable storage, making slow to search a particular signature, and affecting system performance [4]. In addition, it is estimated that at least 24 hours are required for a new signature to be added to the database, giving new variants the ability to compromise a system without being detected. These are the reasons why researchers are working on new proposals for detecting malware [12].

A novel approach for detecting polymorphic malware is discussed in [2]. It is based on identifying files that are dropped when malware is executed. Hash of every dropped file is calculated and compared with other samples to determine if there are any similarities. Although this approach makes a differentiation by hashing algorithms, it can still be considered as signature based method since it generates signatures (hash) of malware samples. This technique also has weaknesses; changing the code of the examined sample examined or changing the name of the dropped files might result in evading such technique.

B. Behaviour based Method

The approach is to identify a malware by inspecting its behavior while it is being executed [4]. A behavior-based detector can determine if a program is malicious or not by examining what it does [3]. The architecture of a behavior-based detector, it consists of:

- Data collector which acquires dynamic and static information from an executable
- Interpreter which transforms collected data to intermediate representations.
- Matcher which compares representations with behavior signatures.

Symantec corporation implements this technique as part of its anti-malware solutions [3].

In spite of this approach focuses on behavior, it might produce a considerable false positive ratio as well as high amount of scanning to achieve its purpose. Behavior based approach relies on identifying patterns that are not common, such data being sent between two nodes, attempts to change boot sector or the flash memory [4], [11].

C. Heuristic based Method

It was born as an alternative to signature-based detection. It works by examining system behaviors and keystrokes to determine if there is an abnormal behavior. Also, it does not require constant updates. But, it can produce several false positives as well as consuming more computing resources. Besides, it requires the inclusion of a third party component, such as tools to analyze protocols which might open a new breach in terms of vulnerabilities. Finally, heuristic detection has to know the vulnerability rather than the malware [4].

It is important to clarify that heuristic approach reviews the code to find possible variants of a malware, while behavior-based method examines if a malware misbehave. The combination of heuristic and behavior methods could help to reduce false positives. On the other hand, excluding the inclusion of a third party component would prevent vulnerability exploitation from other sources. This approach would reduce system performance effect [4].

Heuristic method is considered as part of a technology that uses Artificial Intelligence. This method has the ability of self-discovering and analyzing the code in an intelligent way performing a deep inspection of code instruction sequences as well as discovering unusual or unopened system calls. This technique might cause a high rate of false positives and negatives but combining with other traditional techniques, its efficiency could improve considerably [13].

Some heuristic methods are listed below [3]:

API/System calls: Based on the calls made to the Operating System (OS) by a particular program through the use of application programming interface (API). The aim is to analyze what piece of code uses a request to the OS.

OpCode: It is based on Operational Codes (subdivision of machine language) in order to identify code sequences that might be associated to malware programs. Some of the approaches shown previously, count them and compare with valid programs.

N-Grams: This approach is based on reading binary code either from reading PE section, plain-text strings encoded in executables and sequences of bytes.

Control flow graph: A program is represented through a series of steps; every section is represented by a node and helps to understand the way a program functions.

Hybrid features: This method considers the inclusion of Machine Learning classifiers which depends on two elements: features and algorithms. In order to improve accuracy of this method, it is required to combine features.

D. Artificial Intelligence-based Methods

Artificial intelligence aims to simulate human thinking and behavior processes like learning, planning, among others [13].

The improvement of anti-virus systems has considered the inclusion of artificial intelligence technologies as a way to increase accuracy and performance. Several technologies are

discussed since they are applied in anti-malware detection systems [13].

1) Data Mining

This technology analyzes sets of odd relations and summarize it to make it understandable for the owner of data.

Several data-mining frameworks have been purposed in order to detect new malwares by identifying new patterns as well as strange connection behavior. Indeed, the main purpose of its inclusion is to extract additional data that is not considered in common techniques like signature based and then build new rules for detecting them [13].

2) Agent Technology

It intends to help computers respond automatically whenever a malware strikes. This technology provides a collaboration mechanism that first uses the information of host files instead of malware files. Then, once the virus has been detected, it tries to recover the original files from the local network, it works as a backup system that recovers infected files from clean hosts. This mechanism is shown in Figure 1

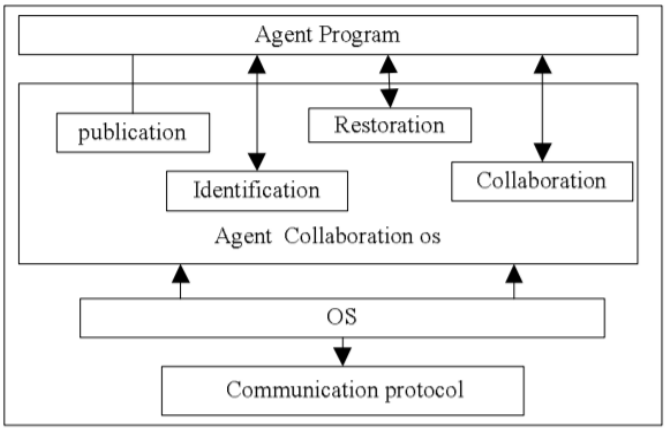

Figure 1: Agent Collaboration Mechanism [13]

3) Artificial Immune Technology

This approach is inspired on biological immune systems that can withstand and destroy biological viruses. Few works have been developed based on this technology. Some of those solutions are (1) a system that combines evolutionary algorithms and agents in order to identify new types of malware, and (2) an immune-associative-memory-based that combines immune first reaction for unknown malware with associative memory idea of second reaction for known malware. The second approach was able to detect unknown, new variants, and similar malwares [13].

4) Artificial Networks

Due to the lack of associative memory and real-time calculation capacity, researchers have considered the use of Artificial Networks to provide parallel process, self-organization, self-learning, and information classification. Back in 1994, IBM used a Single Layer neural classifier to detect boot and generic malware, but, it has not been widely adopted yet. The inclusion of genetic algorithms allows to extract rules in order to classify patterns [13].

Other works are being conducted using Neural Networks for malware classification. Some of them are looking into the code and PE whilst, others are willing to read binary information from executables and converting it to images for extracting malware features using Gabor Wavelet and GIST [14].

E. Sandboxing

This mechanism consists on running programs in a separate/isolated environment by controlling all the resources allocated in case of damage. This is considered an appropriate contention mechanism against obfuscation as discussed by [9]. There are several tools that use this technique. Such tools are able to imitate malware interaction as well as catching and documenting changes made to an infected system [15]. Once a sample of malware is in execution, it is important to collect as much information as possible in order to determine changes performed in the system, identify patterns and understand its behavior.

Currently, this technique could be outsmarted as malware developers are including instructions for detecting a sandboxed environment to prevent malware execution, giving it the ability to run on valid hosts only.

F. Dynamic Analysis

This approach focuses on analyzing malware behavior whilst it is being executed. It is recommended to perform this action over a controlled environment (e.g. virtual machine) together with monitoring tools for checking processes changes in files as well as network monitoring for open connections [16]. This technique is used as part of the malware analysis process as well as the extraction of signatures; however, it could not be considered as a technique for detecting or identifying in early stages. Therefore, this a post-attack technique only and it could not be used as defense mechanism.

MALWARE EVASION TECHNIQUES

As there are ways to detect malware, there are also ways for being undetected; just like a game of thieves and cops. There are several techniques used by attackers that are leaving behind anti-malware vendors [9].

A. Encryption

Encrypted malware is composed of two main sections: a decryption loop and a main body. Decryption loop is capable of encrypting and decrypting the main body. Main body contains the code of the malware itself which is encrypted with simple algorithms like XOR or using complex and robust ones such as AES. Anti-malware solutions must decrypt the main body to get the valid signature and detect the malicious piece of software [17].

B. Oligomorphism

The purpose of this technique is to produce a different decryptor on every new infection. An additional improvement is that there are several decryptors which are randomly chosen making a new type of code on every instance. This technique can be detected but requires more time [17].

C. Polymorphism

Polymorphic malware is harder to detect as there is an unlimited number of new decryptors. The main feature of this technique is that the code constantly changes on every new variant. Code obfuscation is used to mutate the decryptor to produce a new version for another victim [17].

D. Metamorphism

It does not contain an encrypted part. However, it uses mutation engines to change the body on every compilation, rather than using cryptography for protecting the code. A metamorphic engine should consist of the following components as shown in Figure 2:

- Disassembler
- Code analyzer
- Code transformer
- Assembler

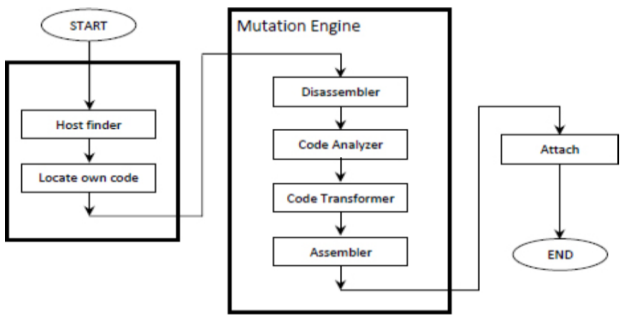

Figure 2: Structure of replicator and mutation engine [17]

To detect this type of technique a complex and robust engine including heuristics and behavior analysis has to be in place. However, as stated in [17] there is no solid approach in place that helps detecting this type of malware.

E. Obfuscation

Hiding information to avoid being caught is a common practice among attackers, in order to defeat certain security devices like IDS or any other based on signature detection. Using encoding and manipulating strings is a practice that can easily bypass Snort signatures. For example, either by replacing a / with a \ or by using Hex, Unicode or UTF-8 can avoid detections [9]. Moreover, using encryption to encode a whole session is a classic way to avoid being detected. The problem lies in the lack of the appropriate key to decrypt information. Polymorphic code aims to mutate its code while maintaining its original algorithm. It is usually combined with a cipher/decipher module that is embedded in the code. Encrypting malware is the first stage to bypass signature-based solutions as they do not have readable information to compare [18]. Likewise, metamorphic malware presents a novel approach as it improves obfuscation, it has to recognize, parse and mutate its own body every time it wants to propagate [17], [18].

Obfuscation has helped polymorphic and metamorphic malware to bypass anti-malware solutions. Indeed, they have used several coding techniques, to achieve its goal, some of them are [9], [17], [18].

Dead-Code insertion: It simply adds not effective instructions to a program to change its appearance but its behavior remains intact.

Register Reassignment: It consists on switching registers from one version to another.

Subroutine Reordering: A subroutine is obfuscated and reordered in a random way giving n chances of variants.

Instruction Substitution: Its objective is to replace original code with others that are equivalent to the original.

Code Transposition: It reorders the sequence of a set instructions from the original code, either by using unconditional branches or based on independent instructions.

Code Integration: A malware joins its code with a valid program by decompiling the original one and rebuilding it with infected instructions.

F. Fragmentation and Session Splicing

This are network attack evasion techniques which take advantage of a feature of IP protocol called packet fragmentation, that allow to handle packets of different sizes. This evasion technique affects security devices as they have to wait for the whole package to arrive and then analyze it [9].

G. Protocol Violations

Weaknesses are present in protocols like RPC which has security issue from its design as it allow other computer to remotely execute commands [9].

H. Code reuse attacks

These type of attacks does not inject code in order to work. First of all, it uses return-into-libc which aims to reuse executable code from a valid running process. A malicious user might try to change the pointer to a different function rather than following the normal process. Libc is a standard library that is part of C-language.

Return-Oriented Programming (ROP) can create malicious computation orders by linking small pieces of code in existing the space address of a running program. These type of attacks are organized in sets of instructions and does not require to inject code or call a function which is a way to bypass current anti-malware solutions [9].

I. GPU-assisted malware

Graphics processing units (GPUs) had been in charge of dealing 2D and 3D rendering tasks. However, lately malwares have started using those resources. In [20] a prototype based on this approach was built. The author uses different techniques such as basic self-unpacking, brute-force unpacking, and runtime polymorphism for deploying the malware. It is important to remark that this type of malware does not run on a virtual machine as it uses specific GPU libraries which are not currently virtualized. On the other hand, GPU has the ability of accessing host memory directly making it shared between CPU and GPU. The author let GPU to perform tasks that might be noticed on the CPU to meet the goal of going undetected [19].

Although this new type of malware uses known packing techniques, it takes advantage of the power given by GPUs to evade current anti-malware solutions. This technique might not be applicable to all devices, but it opens the door for using other computing devices rather than traditional (CPU and RAM).

J. File-less malware

A new type of evasion technique was discovered by Kaspersky Labs recently on February 2017 according to [20]**.** The analysis showed that digital traces of the malware have been left in the physical memory of a comprised server. The report showed that common tools like Metasploit with different techniques can result in a new way of evading anti-malware solutions.

In addition, another report by Airbus Security, posted in 2016, showed that the code was loaded in registry keys which were not able to be retrieved as they include no-ascii characters. Besides, the content of keys contained some sort of obfuscation to increase complexity [21].

Leaving no digital traces on permanent storage seems a simple technique which works for meeting the goal of going unnoticed by anti-malware solutions. Hence, this technique could be considered as part of the payloads to be used for compromising a specific victim.

K. Virtual machine based malware

Virtual machines are used nowadays in several scenarios such as cloud environments, penetration testing labs, malware analysis among others. Virtual machines rely on a Virtual Machine Monitor (VMM) which is in charge of managing resources and exporting hardware-level abstractions to the guest machine by using an emulation software [22].

Virtual Machine Introspection (VMI) is a family of techniques used to understand Virtual Machine (VM) states and events within a guest. It allows a VM service, which is implemented outside of the guest, to invoke services or applications or guest code [22]. It means, that considering this approach (VMI), someone could be able to inject processes over running ones in a Virtual Machine [23]. Indeed, a framework has been developed for achieving such goal considering four security requirements: stealthiness, isolation, robustness and completeness to guarantee kernel integrity. It means that the implanted process would go undetected by other processes in the guest OS (i.e. Antivirus). Also, it could interact with very few processes in the system to avoid making too much noise. Moreover, it could not be terminated by other processes in the guest VM. Finally, it could terminate without disrupting guest OS or running applications [23]. Considering the requirements that are to be met, it could be said that a malware could fulfill them without problems. Indeed, this technique might allow to inject malicious processes into a guest OS without being noticed by an anti-malware solution. Moreover, this technique might be exploited to compromise large infrastructures that rely on virtualization. This scenario opens the opportunity for a malicious user to build rootkits and take control of all the guests VM running over a physical host as showed in [22]. Likewise, compromising a particular process of a single guest outsmarting anti-malware solutions that might be in place in the host OS. It is important to consider that in order for a virtual machine to be protected, anti-malware solution has to be installed as the solution running over the host does not have the capability of reading memory or files residing on a guest VM.

Therefore, taking advantage of the goodness of VMI might allow to build a piece of malware and hence produce a new evasion technique that will target guest VMs.

L. Silent SFX

Self-Extracting Archive is a type of executable file which contains more files that were compressed. This type of file has a security protection which allows the use of a password to protect information inside it and hence avoiding the inspection of anti-virus by hiding such information. Silent SFX inherits the properties mentioned before to protect malware. This technique would not require user input as it may be handled directly by the SFX archive. A malware to be deployed shall contain a script for passing parameters to the insider compressed file (containing malware), and a decryptor. This technique was tested giving a positive evasion effect as only 2 solutions from 54 tested where able to caught its harmful behavior [24].

This technique shows a different approach for hiding malware by taking advantage of compressing files.

M. Anti-Malware evasion techniques on mobile devices

Mobile devices such as smartphones look more like a computer as they have common elements such as RAM, CPU and GPU. Therefore, they are susceptible to become infected by malware. Likewise, malware for such devices use same evasion techniques as malware PC. However, as shown in [25] using techniques like changing the name of a package or repacking an apk are ways to elude anti-malware solutions as those are patterns used to generate a signature for its containment. The author produced a framework for combining several evasion techniques and effectively bypassing tested anti-malware solutions.

MACHINE LEARNING IN MALWARE CONSTRUCTION

Certainly, there is few literature that highlights specific techniques such as [26]. Such document talks about the use of Evolutionary Algorithms and Genetic Programming. Other researches such as [7] proposes a framework for malware evolution based on genetic algorithm. While, reference [8] discusses several types of malware that could appear by exposing features like intelligent, biological, and human like behavior as well as the artificial intelligence capabilities that might be applied for designing countermeasures which are artificial neural networks, expert systems, fuzzy searches, and biological behavior. In reference [8] several types of malware with artificial intelligence are given, one of them is called Zellome which uses genetic algorithms to brute force decryptor routine and improve polymorphic behavior. Besides, Storm is a type of malware which can adapt its defenses letting it stop countermeasures, and Nimda is a powerful worm that clearly adopted a parasite like biological behavior. Additionally, there are malwares that behave like humans e.g., IM.Myspace04.AIM started short communications with several users using slang. Likewise, CyberLover conducted flirtation to extract personal information from victims. Finally, the author in [9] concludes that malware and anti-malware have used Artificial Intelligence and thus the future would be to emulate human characteristics for utilizing in any of the two sites.

The use of machine learning applied to improve malware is discussed [5] and [6]. The first work focuses on using supervised learning whilst the second is oriented to unsupervised learning, both of them using neural networks as a layer that triggers the trojan.

The proof of concept (PoC) described in [5] is shown in Figure 3 and uses a neural network which has been trained but using a compromised dataset plus code that was modified. Also, payloads have been encoded into the weights as well as the application data. The neural network will have the ability to trigger the trojan once it receives a particular set of data. The author, has defined several sets of instructions like initating, terminating or running commands, which have been encoded to evade anti-viruses as they might include commands like netcat (used for opening sockets) or instructions that may reflect anomalous behavior. Finally, a decoder is used in this PoC which is in charge of translating sets of data into trojan.

On the other hand, the neural trojan is unlikey to be detected as the output is encoded and the payload used is part of valid datasets; therefore, it could not be tagged as a malicious application until it has been translated. However, the actions that will be executed might be considered as suspicious since they will perform actions like open files, sockets, among others [5]. Therefore, the discussed approach might be detected at the point where instructions cannot be furthered hidden from the operating system.

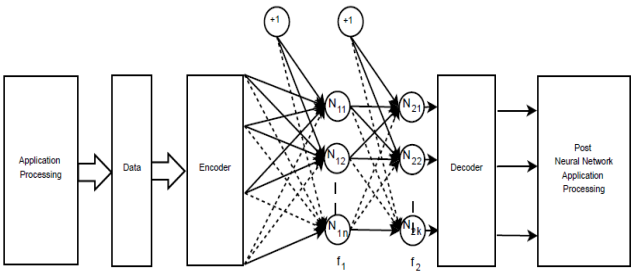

Figure 3: Neural Network Trojan design proposed in [5]

In summary, the aforementioned approach uses a neural network for "ciphering" commands that triggers actions for the trojan. The key of this approach is to train the network with infected datasets (1 and 0s) along with valid information. This technique shows an example of evasion using machine learning might.

The second PoC discussed in [6] analyses the viability of improving the previous approach by using unsupervised learning. Figure 4 shows the basic architecture of an unsupervised learning trojan. This work uses a set of payloads

that does not overlap with benign clusters which have to be encoded before injecting them into benign data. The author states that this scenario gives the attacker full control to malware properties as well as not requiring access to a training set. Although approach shows improvements in regards of payload encoding, still there is a chance of being detected when running instructions from memory as in the previous approach.

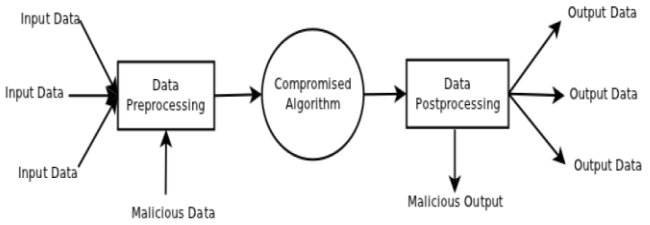

Figure 4: Unsupervised learning architecture algorithm **[6]**

DISCUSSION

Novel approaches were described in previous sections in regards of malware detection. Some of them are looking forward to tag (identify) every piece of malware or even classifying them as part of a family using hybrid techniques. The problem lies in the exponential growth of malwares including zero-day vulnerabilities and lack of human awareness in terms of information security. The approaches presented by the research community does not seem to be hardly supported by anti-malware vendors as there are elusive solutions still in place for detecting and containing malware attacks. Besides, the inclusion of machine learning for contention seems to be hardware demanding, limiting the ability to develop end-user solutions. In addition, having a big dependence of being connected all the time to Internet to receive "updates".

Several evasion techniques also have been described in previous sections. Currently, malwares use one of those techniques no matter the type of the system. However, better approach would be to determine the best, a combination of them or a new one, based on device features without affecting system performance and meeting the goal of evading anti-malware solutions. Machine Learning might be considered as a tool to determine the best conditions. Besides, the target objectives of malwares should also consider Internet of Things (IoT) for extending their field of action with a certain degree of autonomy.

On the other hand, frameworks for building malwares are gaining territory and there are some free to use [27], [28]. They offer generating malwares that bypass common anti-virus (AV) solutions. They additionally offer the feature of "open source" which could be very useful when customizations are required. Finally, ethical hacking distributions also provide frameworks like SET or Metasploit which are commonly used in malware development.

CONCLUSIONS

Although there are several malware detection techniques, they are not perfect ones. It means that they have to be executed together in a hybrid approach to have better performance.

Including machine learning techniques as part of anti-malware solutions has not become an anti-malware industry standard, but simple techniques such as Signature based one are still in place giving malware the potential to bypass them by performing minor changes such as renaming variables or obfuscating code.

Ransomware exploited the benefits of encryption to harm information. A large number of research is being conducted to include machine learning as a mechanism of malware detection. In this aspect, a few papers have been published containing mechanisms to use machine learning as generation engine for malware. In addition, modern evasion techniques that use trending technologies are being explored such as GPU and Virtual Machines. Until now, there is no evidence yet of malware that uses machine learning as a means to determine the best way to infect a system. Therefore, this scenario requires in-depth research to determine the feasibility of this possible type of attack as well as identifying probable ways to stop it.

The use of machine learning for improving malware evasion techniques has not gone beyond using genetic algorithms (GA) to generate new variants (metamorphism). However, there is an approach that uses neural networks as a layer of communication to hide commands. Although that approach does not use the neural network as a classifier to select the best evasion technique or to avoid being detected, there is no restriction to achieve such goal.

Evasion techniques are being documented and exploited in the wild. There are works such as [27], [29], [30] looking for evading anti-malware solutions. In fact, there are frameworks for fingerprinting anti-virus which might be considered for the purpose of our research. Hence, it would provide data of the program to be evaded turning it to an input of the neural network in charge of determining the best type of technique to apply.

Research community is majorly focused on detecting and classifying malware. As a result, several novel techniques were proposed. However, those approaches have not been implemented yet by anti-malware vendors. On the other hand, literature on taking advantage of trending technologies like cloud, GPUs or machine learning is much reduced. But it motivates to perform a deep research in such field to discover new types of malware that might appear.

REFERENCES

[1] "Malware Statistics & Trends Report | AV-TEST." [Online]. Available: https://www.av-test.org/en/statistics/malware/. [Accessed: 25-Aug-2017].

[2] N. S. Selamat, F. H. Mohd Ali, and N. A. Abu Othman, "Polymorphic Malware Detection," in *2016 6th International Conference on IT Convergence and Security (ICITCS)*, 2016, pp. 1–5.

[3] Z. Bazrafshan, H. Hashemi, S. Mehdi, H. Fard, and A. Hamzeh, "A Survey on Heuristic Malware Detection Techniques."

[4] A. A. Sulaiman Al Amro, "A Comparative Study of Virus Detection Techniques," *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 9, no. 6, pp. 1566–1573, 2015.

[5] A. Geigel, "Neural Network Trojan | Arturo Geigel - Academia.edu," *J. Comput. Secur.*, vol. 21, no. 2, pp. 191–232, 2013.

[6] A. Geigel, "Unsupervised Learning Trojan," Nova Southeastern University, 2014.

[7] S. Noreen, S. Murtaza, M. Z. Shafiq, and M. Farooq, "Evolvable malware," *Proc. 11th Annu. Conf. Genet. Evol. Comput. GECCO 09*, p. 1569, 2009.

[8] J. Pan and C. Fung, "Artificial intelligence in malware-Cop or culprit?," pp. 181–184, 2008.

[9] J. a P. Marpaung, M. Sain, and H.-J. Lee, "Survey on Malware Evasion Techniques: State of the Art and Challenges," *14th Int. Conf. Adv. Commun. Technol. (ICACT),* no. Mic, pp. 744–749, 2012.

[10] F. Touchette, "The evolution of malware," *Netw. Secur.*, vol. 2016, no. 1, pp. 11–14, Jan. 2016.

[11] A. S. T. and . B. HERBERT, *MODERN OPERATING SYSTEMS*. New Jersey: PEARSON, 2015.

[12] E. Kuldeep Singh Lakhwinder Kaur, "A Survey of Various Malware Detection Techniques."

[13] X.-B. Wang, G.-Y. Yang, Y.-C. Li, and D. Liu, "Review on the application of Artificial Intelligence in Antivirus Detection System."

[14] A. Makandar and A. Patrot, "Malware analysis and classification using Artificial Neural Network," in *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, 2015, pp. 1–6.

[15] M. Egele and C. Kruegel, "6 A Survey on Automated Dynamic Malware-Analysis Techniques and Tools," 2012.

[16] E. Gandotra, D. Bansal, and S. Sofat, "Malware Analysis and Classification: A Survey," *J. Inf. Secur. J. Inf. Se-curity*, vol. 5, no. 5, pp. 56–64, 2014.

[17] B. Rad, M. Masrom, and S. Ibrahim, "Camouflage in Malware : from Encryption to Metamorphism," *Int. J. Comput. Sci. Netw. Secur.*, vol. 12, no. 8, pp. 74–83, 2012.

[18] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," *Proc. - 2010 Int. Conf. Broadband, Wirel. Comput. Commun. Appl. BWCCA 2010*, pp. 297–300, 2010.

[19] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "GPU-assisted malware," *Int. J. Inf. Secur.*, vol. 14, no. 3, pp. 289–297, 2014.

[20] Kaspersky Lab, "Fileless attacks against enterprise networks," 2017. [Online]. Available: https://securelist.com/fileless-attacks-against-enterprise-networks/77403/.

[21] B. A. Dove, P. Behavioural, A. Fileless, M. Kovter, and P. Bot, "Fileless Malware – A Behavioural Analysis Of Kovter Persistence," pp. 3–6, 2016.

[22] S. T. King, P. M. Chen, Y. M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch, "SubVirt: Implementing malware with virtual machines," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2006, pp. 314–327, 2006.

[23] Z. Gu, Z. Deng, D. Xu, and X. Jiang, "Process implanting: A new active introspection framework for virtualization," *Proc. IEEE Symp. Reliab. Distrib. Syst.*, pp. 147–156, 2011.

[24] S. Bhushan, P. Kumar, A. Kumar, and V. Sharma, "Scantime antivirus evasion and malware deployment using silent-SFX," *Proc. - 2016 Int. Conf. Adv. Comput. Commun. Autom. ICACCA 2016*, pp. 5–8, 2016.

[25] V. Rastogi, Y. Chen, and X. Jiang, "Catch me if you can: Evaluating android anti-malware against transformation attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 1, pp. 99–108, 2014.

[26] A. Cani *et al.*, "Towards Automated Malware Creation : Code Generation and Code Integration," pp. 157–158, 2014.

[27] A. Experiment and A. V Evasion, "peCloak . py – An Experiment in AV Evasion," no. January, pp. 1–16, 2015.

[28] Chris Truncer, "Veil-Evasion - Veil - Framework." [Online]. Available: https://www.veil-framework.com/framework/veil-evasion/. [Accessed: 25-Aug-2017].

[29] J. Blackthorne, "AVLeak: Fingerprinting Antivirus Emulators Through Black-Box Testing," *Usenix Woot*, 2016.

[30] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artifacts," *S&P*, pp. 1009–1024, 2017.