Antivirus Software: How Does It Work ?

Rayan Hassan
Swanson School of Engineering
University of Pittsburgh
Pittsburgh, United States
rah194@pitt.edu

Catrina Wolf
Swanson School of Engineering
University of Pittsburgh
Pittsburgh, United States
caw190@pitt.edu

Abstract - Antivirus software is important as they protect us against malware and cybercriminals. It works by scanning incoming files or code that's being passed through your network traffic. Companies who build this software compile an extensive database of already known viruses and malware and teach the software how to detect, flag, and remove them. Antivirus software also protects us against keyloggers, worms, rootkits, etc…

Keywords - antivirus, malware, signatures, polymorphism, string, heuristics, mutation, crossover, population, virtual machine, dynamic analysis, static analysis, sandbox, machine learning, behavior monitoring, zero-day malware, fileless malware, operating systems, Windows Defender, XProtect, Gatekeepers, Linux

I. INTRODUCTION

Antivirus is a kind of software used as an additional layer of protection that once installed, runs automatically in the background to secure devices against viruses. It searches for, detects and finally removes viruses. Some “antivirus” software further extend to prevent other attacks; these are called “anti-malware” software. Although these two words don’t technically have the same meaning, they are often used interchangeably these days. Malware is a general term that refers to any kind of software attack (worms, ransomware, Trojan horses, etc…), whereas virus is a form of malware. It is specifically a program that replicates itself by damaging other programs. For generalization purposes, we won’t be distinguishing between them, as they both follow the same working scheme, with only a few differences. Signature analysis, heuristic analysis, sandbox detection, machine learning and behavioral monitoring are mainly used to combat them. We will be discussing each one of these operations in details, as well as taking a general look at what antiviruses modern operating systems use to prevent malware attacks.

II. SIGNATURE ANALYSIS

Antiviruses use signature-based detection to prevent attacks. Each malware has a unique footprint which is stored in a database. The program then searches for footprints that match those of known malware. If found, it detects and removes it. Every time a new type of malware is discovered, its footprint is added to the database [1]. To generate these footprints, security analysts gather multiple samples of the malware file and study their common attributes like size, data bytes at certain positions, hashes, etc… Signatures are then generated following a specific format. The most commonly used format is YARA, which creates them based on textual and binary patterns. Fig.1 shows a rule that consists of a set of strings and Boolean expression that describes a malware family [2].

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

Fig.1. Example of a silent banker rule that follows the YARA documentation [2]

It indicated that any file containing one of these three strings ($a, $b or $c) can be reported as a silent banker, which is a form of Trojan horse that steals credential and confidential information when a user tries to log in to their bank account. This is a simple example, other rules can be more complex and use other features that are more advanced. Another malware signature is shown in Fig.2.

Fig.2. A sample of OceanLotus malware and its detection signature [2]

OceanLotus is a backdoor attributed to a Vietnamese Advanced Persistent Threat (APT). A backdoor is a type of malware that prevents authentication procedures to access a system. The conditions in Fig.2 specify that the file must have a size less than 200KB, and follow the Mach-O format, which is basically a binary stream of bytes grouped into chunks of data [2].

Signature-based detection has many advantages since it can target whole families of malware rather than just one sample, since analysts study multiple samples of the file. Also, it can detect a wide range of file-based malware. However, it doesn't prevent hackers from stepping ahead of it. In fact, a common strategy is to mask malware from being discovered. One way of doing that is to write a polymorphic code which mutates while keeping the original algorithm intact [3]. That way computer viruses and worms can hide their presence. For instance, 3+7 and 11-1 have the same answer, but run with different machine code in a CPU. A clearer example is presented in Fig.3.

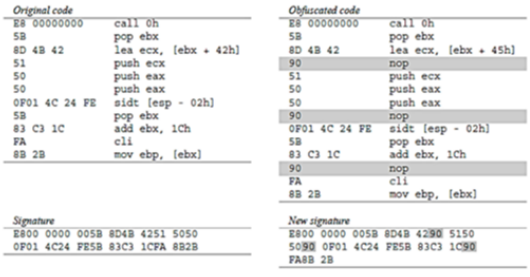

Fig.3. Example of a polymorphic code [5]

This figure compares an original code (on the left) with its "masked" version (on the right). As observed, bubbles (nop) are added in between instructions, which will change the running time down the pipeline on a CPU at the hardware level. More importantly, it changes the machine code and therefore generates a different signature in order for the attacker to hide their malware.

More advanced codes are also employed like metamorphic viruses, which encrypt or modify themselves so that their signatures don't match those of a known virus, similar to polymorphic codes. The main difference is that they can rewrite their own codes. With all these methods, signature-based detections are more vulnerable, but they are still important in antivirus software.

III. HEURISTIC ANALYSIS

Another operation applied in antiviruses is heuristic analysis. This method is designed to detect previously unknown computer viruses and defend against new variants that have not yet been defined [4]. The first step is examining a code for suspicious properties. It analyzes its destination and intent. If it's purpose is to modify or delete certain files, then it can be flagged as a virus. Particularly, a heuristic evaluation is thoroughly run in order to find similarities between the suspicious file and other known viruses. For that, the Longest Common Subsequence (LCS) algorithm is used on all sequences in a set of strings [5]. As the name indicates, it will find the longest common subsequence, which can help detect similarities with a virus. Fig.4 gives a LCS example.

Fig.4. Example of the Longest Common Sequence algorithm applied [5]

Note how this algorithm finds a sequence of characters that is present in the same continuous order in strings A and B. This algorithm is extremely useful as it can detect similarities between files even if they are different, and turns an accurate match by removing the elements that were added through polymorphism [5]. Another algorithm is the Longest Common Substring (LCSubstr), which first starts by finding the Longest Common Suffix (LCSuff) for all

pairs of prefixes in a string [5]. Then, the longest common substring of A and B is found, among all the possible prefixes. Fig.5 shows an example of LCSubstr.

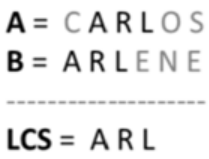

Fig.5. Example of the Longest Common Substring algorithm applied [5]

This algorithm helps find consistent patterns, sometimes even exact token matches that will indicate very high probabilities of polymorphism.

Another heuristic technique is to throw the file in a controlled virtual environment (or sandbox) and monitor its behavior, looking for any suspicious conduct like self-replication or overwriting files. The suspicious program is then decompiled, and its machine code analyzed. This concept will later be discussed in detail.

In order to detect different variations of a virus, genetic algorithms are used. This concept is based on Darwin's evolution theory which states that every creature evolves from an ancestor over a period of time [5]. The genetic algorithm starts with a set of randomly generated possible solutions called chromosomes that form a population, they can be either binary or decimal values. Each one of them has a score, which represents their fitness. This score determines the ability of each chromosome in solving a certain problem. After that, a mating pool is created from individuals based upon their fitness value. The best solutions are taken and modified through random mutations and crossovers in hopes of getting a better population than the old one. Crossover is the process in which the strings of two individuals are mingled together, creating new individuals (children) which have properties of both parents. Mutations on the other hand happen when children are already created. A slight change occurs to ensure that the populations are not identical, like flipping a bit. Finally, the new generation is born and consists of children that have fitness values which are determined. These values are compared with the old population's ones and the individuals that have the highest scores from both populations are chosen to develop a new generation. Therefore, the same process is repeated multiple times. Hence, this algorithm helps find new variants of a virus and detect its signature, preventing it from propagating and causing trouble.

Although heuristic analysis is helpful in many ways, it can still give false positives, which creates a chain of inaccurate results. In fact, if during the first heuristics of a token matching (LCS), a false positive occurs, the file that continues to the evolution stage will give birth to undesired children. This will reduce the system's efficiency and accuracy.

IV. SANDBOX DETECTION

Another system for malware detection is sandbox technology. A sandbox system functions by running a suspicious object in a virtual machine (VM) with a fully-featured operating system to analyze its behavior in order to detect any malicious activity. If the object performs any malicious actions while being run in the VM, the sandbox will detect this as malware. The sandbox functions as a safer, isolated space to run suspicious objects and detect malicious activity safe from any real business infrastructure where there would be risks [6].

Sandboxes allow for dynamic analysis of an object as it executes, making it effective at detection of malware that may escape forms of static analysis [6]. This dynamic analysis occurs by studying the behavior of a malicious code, or its interaction with the system, while it is being executed in a controlled environment. Before executing the malware sample, the appropriate monitoring tools are installed and activated within the sandbox. These include tools for file system and registry monitoring, process monitoring, network monitoring and system change detection [7]. There are also various techniques that can be applied to perform dynamic analysis. These include function call monitoring, function parameter analysis, information flow tracking, instruction traces and autostart extensibility points, etc. [8]. Dynamic analysis is generally more effective and does not require the executable to be disassembled compared to static analysis. It is able to disclose the malware's

natural behavior, which is more resilient to static analysis. However, it is also time intensive and consumes more resources to perform, creating scalability issues [7].

The malware detection process of the sandbox begins with the sandbox receiving a request to scan an object, in the form of a file or a URL. This request includes instructions on the OS and configuration for running the object, its execution parameters, other third-party applications installed in the VM, and any other important variables such as how long to run the test. The object is run in the sandbox according to the specified instructions and the sandbox collects artifacts. Artifacts include key components such as application execution logs (API function calls with their parameters, execution events), memory dumps, any changes in the file system or registry, network traffic, and artifacts of exploit activity among others. The sandbox will collect and analyze the artifacts and send a response to the requesting system, determining whether the object is benign or malware. This verdict is sent with the object's data along with a detailed description of any suspicious activities recorded during the sample's execution. This can help in further analysis without a new request needing to be sent to the sandbox [6].

When dealing with modern malware, it is common for the malware to attempt to detect and evade a sandbox. If the malware knows it is running in a sandbox, it may skip performing malicious activity or even erase itself from disks or terminate itself. The malware may also employ other evasion techniques in response to being run in a sandbox. Some functions and design aspects of certain sandboxes may leave traces that indicate that a suspicious process is being monitored, which the malware would be able to detect. The sandbox can implement other non-intrusive monitoring techniques that leave no visible trace to the scanned object. This can be done by avoiding modifying process operation, memory, and system libraries on disk and in memory, thereby leaving no traces of monitoring activity for the malware to detect [6].

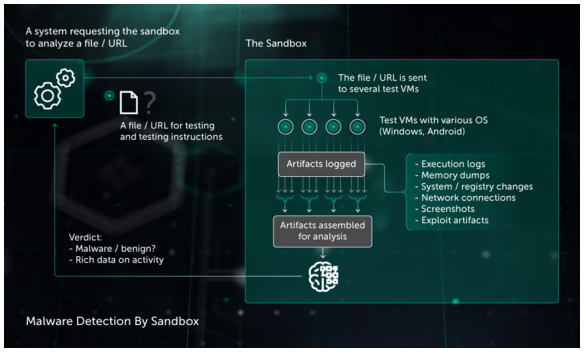

Fig.6. Visual overview of malware detection by Sandbox [6]

V. MACHINE LEARNING

Machine Learning is a form of automated learning by using examples and experience, without being explicitly programmed. Due to the advantages it offers, machine learning techniques are gaining popularity in the field of malware detection. Not only will they detect known malware, but they will also deliver knowledge for the detection of new or obfuscated ones [9].

There are several kinds of machine learning algorithms such as supervised learning, unsupervised learning, reinforcement learning, etc. Supervised learning maps an input to an output based on examples of input-output pairs provided to the machine learning algorithm. It then creates a function from this set of labeled training examples to predict labels for real data. In contrast, unsupervised learning does not use a training set to teach the algorithm how to predict labels for data. Instead, unsupervised learning algorithms are left on their own to discover structures in data. These algorithms function by learning strong features of the data, and grouping data into different categories – a process known as clustering. When new data is introduced, the algorithm uses these previously learned features to determine the class of the data. Reinforcement learning is a machine learning training method based on the notion of rewarding desired behaviors and punishing undesired ones. In this way, a reinforcement learning agent can interpret its environment and determine what actions to take based on trial and error [10].

In malware analysis, there are two phases to machine learning. The first phase is the training one, where the chosen machine learning algorithm mathematically formalizes the set of features extracted from the known malicious and benign data to build a predictive model. These features can be extracted by static analysis, without executing the executables or through dynamic analysis, during the execution of executables. The machine learning algorithm builds a predictive model from the training data to be used to categorize unknown data. The second phase is the testing phase, where the predictive model processes the properties of unknown data to predict whether it is malware or benign [10].

Decision trees are graphs used to represent choices and their results. Each tree consists of nodes and branches. The tree is formed by starting with a root node at the top and having subsequent decisions form the rest of the tree. Each node represents attributes in a group and corresponds to an event or choice. Each branch represents a value that the node can take, with the edges of the graph representing the decision rules or conditions. In this way, data will be classified using the decision rules at each node, starting from the top, until a final classification decision is made by reaching one of the outermost branches of the tree [10]. In a decision tree ensemble, the predictive model consists of a set of decision trees. During the test phase, the model traverses the trees and at the final stage, the decisions of multiple trees are averaged based on an algorithm to provide a final classification decision on the object [11].

Machine learning algorithms are not safe from the threat of attacks, however. These attacks typically aim to force machine learning systems into making deliberate errors, misclassifying the data. An attacker could poison a training dataset to alter the machine learning algorithm in their favor, or reverse-engineer the model's code. It is also possible for an attacker to brute force machine learning models using specially developed Artificial Intelligence (AI) to automatically generate many attacking samples until a weak point can be discovered in the model [11]. In this way, a malicious adversary can manipulate the input data, allowing them to exploit specific vulnerabilities of learning algorithms in order to compromise the security of the entire system [9]. Because of this, security vendors should be aware of machine learning model vulnerabilities and meet requirements for machine learning performance that include a robustness to potential adversaries. Machine learning methods should be used in coalition with complementary protection technology and human expertise as part of a multi-layered security approach [11].

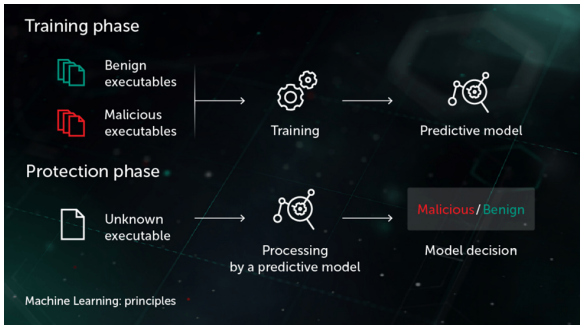

Fig.7. Visual overview of the phases of machine learning [11]

VI. BEHAVIORAL MONITORING

Behavior based detection is one of the most efficient ways to protect against advanced threats such as fileless malware, ransomware and zero-day malware. Fileless malware is a type of malware that uses native, legitimate tools built into a system to execute a cyber attack. Ransomware is a form of malware designed to encrypt files on a device, causing the files and any systems that rely on them to become unusable. Zero-day malware is malware that exploits unknown and unprotected vulnerabilities that are typically recently discovered vulnerabilities [12].

Behavior monitoring uses various runtime detection methods to intercept threats that cannot be detected before execution. Two types of these detection methods are malicious and suspicious behavior detection as well as buffer overflow detection. Suspicious behavior detection dynamically analyzes the behavior of all programs running on the computer to detect and block activity that appears to be malicious. This detection watches all system processes for signs indicating active malware. Signs of malware presence could include suspicious writes to the registry or file copy actions that could allow a virus to run automatically when the computer is restarted. Depending on the detection system, it can

be set to warn the administrator as well as block the process once suspicious behavior is detected. By dynamically analyzing all programs running on the computer, malicious behavior detection can detect and subsequently block any activity that is known to be malicious [13].

Buffer overflow detection is an important aspect of handling zero-day exploits in behavioral monitoring. Similar to suspicious behavior detection, buffer overflow detection dynamically analyzes the behavior of all programs running on the system to detect when attempts are made to exploit a process that is running using buffer overflow techniques. Buffer overflow detection is able to catch attacks targeting security vulnerabilities in both operating system software and applications [13].

Behavior monitoring based detection analyzes the actual process activity in real time and reveals the malicious nature of malware when it comes to the execution stage. Once detected, a system can flag an alarm and terminate the process and perform a rollback of the changes that were made. With packed ransomware, the malware could attempt to find, modify, or delete important files on a target system, and this would be enough information to be detected by behavioral monitoring. In some cases, behavior based detection technology is the only way to detect and protect against forms of fileless malware. When making judgements based on behavior-based monitoring, quickly producing detection of malicious activity is key in preventing any final user's data loss [12].

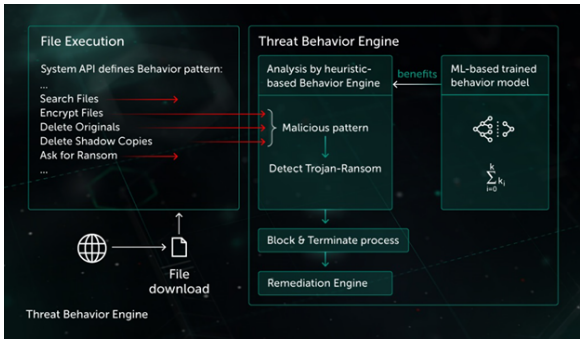

Fig.8. Visual overview of a behavioral monitoring system [12]

VII. MODERN OPERATING SYSTEMS

As more techniques are developed to crack antivirus software and carry out attacks, it is only fair to ask how operating systems deal with the issue. They all have their own built-in antivirus software that takes care of protecting their devices. Let's start with Windows 10, the most used operating system in the world, and therefore the primary target for malware. Windows 10 has a pre-installed and pre-built antivirus software called Windows Defender. It generally is better than most antiviruses, as it uses sandboxing tools and cloud-based malware detection. Number two on the list is Mac, another widely known operating system. Although it doesn't come close to Windows 10 in terms of popularity, it is still used by a large audience. It has two features called XProtect and Gatekeeper. The first works by scanning every app individually for any signs of trouble. The latter makes it impossible for anyone to run any software that isn't created by a certified developer, which further helps prevent malware. Next is Chromebook, another moderately used system. It has a different approach when it comes to dealing with malware. In fact, Chromebook uses Linux as an open-source operating system. It has several advantages like faster performance and more organized file systems. Additionally, Chromebooks have built-in features like sandboxing tools that keep every app and web page separate from the rest of the computer to avoid the spread of any issue. Finally, we have mobile devices, which mainly include Android and iOS. These are much safer from viruses than PCs and laptops since they are primarily devoted to apps. In fact, the user downloads applications from the Google play store for Android, and the App store for iOS. The latter doesn't even allow users to download apps containing malware from any other website. It also isolates every app so that viruses don't spread from one device to another. Android on the other hand doesn't have these restrictions, and users can download apps from untrusted sources at their own responsibility. Android devices also use Chrome as their default browser, which has built-in security protections. In general, it is always better to use another antivirus software as it might protect devices from additional attacks. Some of the most efficient, downloadable antiviruses include ESET, Norton, Bitdefender and McAfee.

VIII. CONCLUSION

In conclusion, anti malware software is a tool that protects devices from all sorts of attacks. Specifically, antiviruses take care of viruses that propagate along a system and replicate themselves. Many techniques are used to ensure a system's security, like detecting a corrupted file by comparing its footprint to the ones of known viruses or malware. Anti-malware software goes beyond just detecting and removing existing forms of attack, it is able to fight new, unknown viruses using heuristics and sandboxing tools. In fact, by finding similarities with known viruses or monitoring their behavior in a VM, predictions can be made. These newly discovered viruses or attacks are added to the dataset, which means more protection against a wider range of threats. Machine learning is used inevitably, as systems learn to face these attacks from previous experiences and take the appropriate action. Nowadays, modern operating systems have their own pre-built antimalware software, which provides default protection for users.

REFERENCES

[1] *What is signature-based malware detection?* Logix Consulting Managed IT Support Services Seattle. (2021, January 27). Retrieved April 9, 2022, from https://www.logixconsulting.com/2020/12/15/what-is-signature-based-malware-detection/#:~:text=Antivirus%20products%20use%20signature%2Dbased,the%20footprints%20of%20known%20malware.

[2] *What is a malware file signature (and how does it work)?* SentinelOne. (2021, October 6). Retrieved April 9, 2022, from https://www.sentinelone.com/blog/what-is-a-malware-file-signature-and-how-does-it-work/

[3] Staff, B. (2018, December 14). *Why relying on antivirus signatures is not enough anymore: Webroot*. Webroot Blog. Retrieved April 9, 2022, from https://www.webroot.com/blog/2012/02/23/why-relying-on-antivirus-signatures-is-simply-not-enough-anymore/

[4] *What is heuristic analysis?* Forcepoint. (2021, May 5). Retrieved April 9, 2022, from https://www.forcepoint.com/cyber-edu/heuristic-analysis#:~:text=Antivirus%20heuristic%20analysis%20helps%20software,added%20to%20virus%20definition%20files.

[5] Nasillo, C. (n.d.). *Carlosnasillo/hybrid-genetic-algorithm: Anti-virus research on using heuristics and GA's to evolve and detect new polymorphic virus signatures*. GitHub. Retrieved April 9, 2022, from https://github.com/carlosnasillo/Hybrid-Genetic-Algorithm

[6] Kaspersky. "Sandbox." *Kaspersky Cyber Security Solutions for Home & Business*, https://www.kaspersky.com/enterprise-security/wiki-section/products/sandbox#:~:text=A%20sandbox%20is%20a%20system,sandbox%20detects%20it%20as%20malware

[7] Talukder, Sajedul. *Tools and Techniques for Malware Detection and Analysis*. Edinboro University, 17 Feb. 2020, https://www.researchgate.net/profile/Sajedul-Talukder/publication/339301928_Tools_and_Techniques_for_Malware_Detection_and_Analysis/links/5e4a46e592851c7f7f40fa87/Tools-and-Techniques-for-Malware-Detection-and-Analysis.pdf

[8] C. Raghuraman, S. Suresh, S. Shivshankar, and R. Chapaneri, "Static and dynamic malware analysis using machine learning," in First International Conference on Sustainable Technologies for Computational Intelligence. Springer, 2020, pp. 793–806.

[9] Handa, Anand, et al. "Machine Learning in Cybersecurity: A Review." *Wires Online Library*, 11 Jan. 2019, https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1306

[10] Mahesh, Batta. "Machine Learning Algorithms - A Review." *Research Gate*, International Journal of Science and Research (IJSR), 2019, https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf

[11] Kaspersky. "Machine Learning in Cybersecurity." *Kaspersky Cyber Security Solutions for Home & Business*, https://www.kaspersky.com/enterprise-security/wiki-section/products/machine-learning-in-cybersecurity

[12] Kaspersky. "Behavior-Based Protection." *Kaspersky Cyber Security Solutions for Home & Business*, https://www.kaspersky.com/enterprise-security/wiki-section/products/behavior-based-protection

[13] Sophos Endpoint Security and Control. *About Behavior Monitoring*, https://docs.sophos.com/esg/endpoint-security-and-control/10-6/help/en-us/esg/Endpoint-Security-and-Control/concepts/What_is_HIPS.html