2018 IEEE 6th International Conference on Future Internet of Things and Cloud

# The World of Malware: An Overview

Anitta Patience Namanya
Faculty of Science and Technology
Ankole Western University,
Sheema, Uganda.
anamanya@awu.ac.ug

Andrea Cullen, Irfan U. Awan
School of Electrical Engineering
and Computer Science, University
of Bradford.
Bradford, UK
a.j.cullen,i.u.awan@bradford.ac.uk

Jules Pagna Disso
Research and Development,
Nettitude Intelligent Cyber Security
and Risk Management.
Leamington Spa, UK
jpagnadisso@nettitude.com

*Abstract*— **Malware, short for malicious software is a program code that is hostile and often used to corrupt or misuse a system. Introducing malware into a computer network environment has different effects depending on the design intent of the malware and the network layout. Malware detection and prevention systems are bypassed by malicious files in computer systems as malware become more complex and large in numbers. This paper presents an overview of the world of malware with the intent of providing the underlying information for the intended study into developing malware detection approaches.**

***Keywords: Malware, Malware evolution, Malware Analysis, Malware detection.***

## I. INTRODUCTION

The internet and the worldwide web have given great advancement in how society communicates and the face of business. This has also given rise to the number of propagation avenues available to malware. Introduction of malicious code in one node can create a chain reaction across all the nodes accessible through the network the node seats on as seen in the popular WanaCrypt0r ransomware. With organisations and countries heavily relying on network technologies, the commercial value of computer networks implies that exploitation of the vulnerability of the business networks can cripple its operations and access to intellectual property and personal information can be acquired by cyber criminals. This creates a commercial opportunity for malware and anti-malware ventures and therefore it is no surprise that it is estimated that cybercrime is expected to raise [1].

It is always a game of "catch me if you can" with malware writers deploying new techniques to thwart the devised analysis and detection in malware solutions [2]. The anti-malware community continue to build systems that are targeted at fighting malware as a way of ensuring that our cyberspace is more resilient while malware use various techniques to survive as long as possible.

## II. MALWARE OVERVIEW

Malware is defined as "any code added, changed or removed from a software system in order to intentionally cause harm or subvert the intended function of the system" [3]. The fact that malware can cause loss of information, money as well as life represents a big threat to technology advancements. The classification of malware depends on execution characteristics of the program. Malware is also classified depending on its payload, how it exploits or makes the system vulnerable and how it propagates [4]. This enables the malware to be subdivided into different types as discussed below.

*Virus*

A virus is self-replicating malicious program. It exists as an executable and spreads by copying itself to other host systems. It is passive and needs to be transferred through files or media files or network files. Depending how the complex the code is, it can modify the replicated copies of its self [7], [8]. Viruses can be used to harm host computers and networks, steal information, create botnets, render advertisements, and steal money among other malicious activities.

*Worm*

This is a self-replicating and active malicious program that can spread over the network by exploiting various system vulnerabilities. It uses targeted vulnerabilities in the operating system or installed software. It contains harmful routines but can be used to open communication channels which serve as active carriers. The Worm consumes a lot of bandwidth and processing resource through continuous scanning [5] and makes the host unstable which can sometimes cause the system to crash. It may also contain a payload that are pieces of code written to affect the computer by stealing data, deleting files or create a bot that can lead the infected system being part of a botnet [7]. While viruses require human activity to spread, worms have the ability spread and replicate independently.

*Trojan horse*

Commonly referred to as Trojan, this is a program that presents as a legitimate software which when downloaded and executed embeds malicious routines or files on the host [7]. In most cases, the Trojan horse when executed will install a virus or may have no payload. It cannot self-replicate and relies on the system operators to activate. It can however give remote access to an attacker who then can perform any malicious activity that is of interest to them. Trojan horse programs have different ways they affect the host depending on the payload attached to them and are usually spread through social engineering [8].

*Spyware*

978-1-5386-7503-8/18/$31.00 ©2018 IEEE
DOI 10.1109/FiCloud.2018.00067

This is a malicious program that uses functions in an operating system with the intention of spying on user activity. They sometimes have additional capabilities like interfering with network connections to modifying security settings on the infected system. They spread by attaching themselves to legitimate software, Trojan horse or even taking advantage of known software vulnerabilities. Spyware can monitor user behaviour, collect keystrokes, internet usage habits and send the information to the program author [9].

*Adware*

Adware which is short for advertising supported software, automatically delivers advertisements seen especially in website pop-up ads and displayed by software. Most are designed to serve as revenue generating tools by advertisers. Some adware may come packaged with spyware which then makes this very dangerous as it can track user activity and steal user information [10].

*Root Kit*

This is a program that employs a set of tools to avoid detection in a system. The tools are very advanced and complex programs written to hide within the legitimate processes on the computer infected therefore are very invasive and are difficult to remove. They are designed with the capability of taking full control of the system and gaining the highest privileges possible on the machine among other possible malicious activities [11]. Because of the evasion techniques used by rootkits, most security vendor solutions are not effective in detecting and removing them and therefore, their detection and remove rely heavily on manual efforts. These may include but are not limited to monitoring computer system behaviour for abnormal activities, storage dump analysis and system file signature scanning.

*Bots*

Bots are programs designed to perform specific operations. Bots are derived from 'robots' which were first developed to manage chat channels of IRC- Internet Relay Chat a text based communication protocol that appeared in 1989 [11]. Some bots are used for legitimate purposes like video programming and online contest among other functions. Malicious bots are designed to form botnets. A botnet is defined as a network of host computers (zombies/bot) that is controlled by an attacker or botmaster. Bots infect and control other computer which in turn infect other connected computers thus formulating a network of compromised computers called a botnet. Bots are very commonly used as spambots, for DDOS attacks, webspiders to scrape server data and distributing malware on download sites. CAPTCHA tests are used by websites to guard against bots by verifying users as humans [11].

*Ransomware*

Ransomware is a program that infects a host or network and holds the system captive while requesting a ransom from the system/network users. The program normally encrypts the files on the infected system or locks down the system so that the users have no access. It then displays messages that force the users to pay to have access to their systems again.

Ransomware use the same propagation means as a computer worm to spread and therefore user awareness and system updates are important mitigation measures as seen in the WannaCrypt0r and Petyr ransomware attacks.

## III. Evolution of Malware

Malware has evolved from the days when it was an exciting prank/experiment gone wrong or uncontrolled to now when malware is used for commercial gain. There are various documented instances of malware created within a laboratory setting like the 1962 Darwin game, 1971 Creeper, 1974 Rabbit Virus and 1975 Pervading Animal [5]. However, all the mentioned malware were kept within a laboratory environment and never escaped to the wild. The first virus known to have been able to escape its creation environment was the Elk Cloner introduced in 1981, six years after the first personal computers [12]. After the success of this prank gone wild, Brain the first Microsoft PC virus was seen in the wild in 1986 [5] and like Elk Cloner, it was more annoying than harmful. However, it is the first virus known to conceal its existence on the disk thus evading detection. The next malware that changed the propagating properties of malware is the Morris worm written in 1988 as an experimental, self-propagating, self-replicating program which was released on the internet [12].

In 1990, Yisreal Radai coined the term malware, short for malicious software that would thereafter be used to as a generic umbrella term for all software with undesired intent within a system [13]. The following decades saw an evolution in malware that is best defined as a two-dynamic evolution; the growth in complexity and malware sample numbers.

The growth in complexity is defined by the different generations of malware seen over the years [14]:

- The first generation (DOS Viruses) of malware mainly replicate with the assistance of human activity.
- Second generation malware self-replicate without help and share the functionality characteristics of the first generation. They propagate through files and media.
- Third Generation utilise the capabilities if the internet in their propagation vectors leading to big impact viruses.
- Fourth Generation are more organisation specific and use multiple vectors to attack mainly anti-virus software or systems due to the commercialisation of malware.
- Fifth Generation is characterised by the use of malware in cyberwarfare and the now popular malware as a service.

Each jump in generation is characterised by increase in complexity of the malware and more propagation vectors. Tricks of the older generation of malware are always seen to be re-utilised in newer generations of malware and complexities discovered over the years always seem to follow the evolving trends in technology [5]. The commercial value attached to having access to exploited systems or the ability to infiltrate a network has led to the birth of malware samples which are very evasive. Some of the most damaging and famous malware of recent times [12], [13], [15], [16] are listed in Table I:

TABLE I. SOME OF THE MOST DAMAGING MALWARE OF THE LAST DECADE

| Year | Malware Name | Details |
|---|---|---|
| 2007 | Storm | Infected about 10 million computers in 9 months |
| | Zeus | Suspected to have infected over 3.6 million computers in US alone by end of 2009. Was later used to distribute CryptoLocker ransomware and became the malware writing kit for malware seen in the later years like GameOver Zeus, Spyeye. |
| 2008 | Koobface | Targeted Myspace and Facebook users |
| | Conficker | Infected over 15 million Windows systems and Microsoft set a bounty of $250,000 on the malware writer. |
| 2010 | Stuxnet Worm | Mainly affected the Iranian nuclear plants and a uranium enrichment plant. |
| 2013 | Cryptolocker Trojan | One of the first ransomware seen where it used social engineering to trick users into infecting their systems. |
| 2013 | ZeroAccess Botnet | Infected over 1.9 million computers and used their resources for bitcoin mining. |
| 2014 | Sony Picture Hack | Sony's film division systems were brought down for two hours and then wiped of data while rebooting. |
| 2016 | Mirai | First malware to scan the Internet of things vulnerable devices and used them to perform DDoS attacks on various sites. |
| 2017 | WannaCrypt0r Ransomware | Globally targeted Windows systems and demanded payment in Bitcoin. Affected over 300,000 machines across 150 countries. |

The evolution of the malware sample numbers is best reflected by the number of malware samples collected by AV-test institute over the last decade as shown in Figure 1 [17]. From the trends seen in the graph, although the total malware samples collected are growing in number, the percentage of the total that is new malware shows a decreasing trend over the last 3 years. This percentage shift enables us to predict that if old malware can be detected and eliminated from the samples automatically, then the analysis time spent on the discovered samples can be greatly reduced.

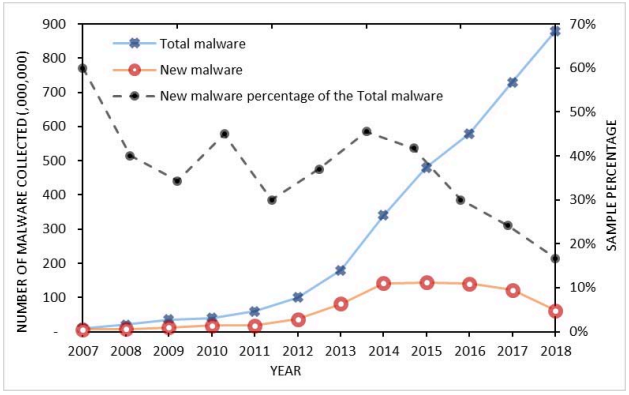


Figure 1. Malware sample statistics for the last decade.

## IV. MALWARE ANALYSIS TECHNIQUES

Malware analysis describes how information about a malware sample is gathered. When a malicious sample is discovered in the wild or on a machine, it is usually an executable which has been compiled and therefore presented in machine language [8]. The main goal of malware analysis is to extract as much information from the discovered sample to understand the threat associated. This allows the security teams to contain the damage, reverse it where possible and build a method to guard systems against future infection by the same type of malware [11]. There are two types of analysis; static and dynamic which can be carried out at a basic level or advanced level based on the tools and methods used. These two can be combined in various stages of malware analysis for optimum result. Recent advancements in malware analysis techniques and tools lead us to adopting and defining malware analysis based a combination of the 4 stages of malware analysis proposed by Zeltser [18] shown in Figure 2. The skill level required in the analysis stages increases with increase on the vertical axis. This multistage analysis allows for the analyst to stop at any of the four stages if they can make a conclusive decision on whether the file is malicious or benign. Its signature can then be modelled to be used in future to detect similar malware.

### A. *Fully Automated Analysis*

Today, the landscape of malware analysis has evolved due to the development and release of opensource, online and or readily available automated malware analysis tools. Automated static analysis tools like Peframe, Pyew and Mastiff provide sample analysis reports very easily that can help a malware analyst with the much needed initial results. Automated dynamic analysis tools like cuckoo, Anubis and ThreatExpert provide malware analysis reports on submission of the malware sample without the need for a laboratory. However, some of the reports require specialised skill and knowledge to full comprehend and therefore can be quite limiting to a user. Then there are hybrid automated tools like VirusTotal [21] which uses over 50 antivirus engines to analyse uploaded malware samples and provides a free report after analysis.

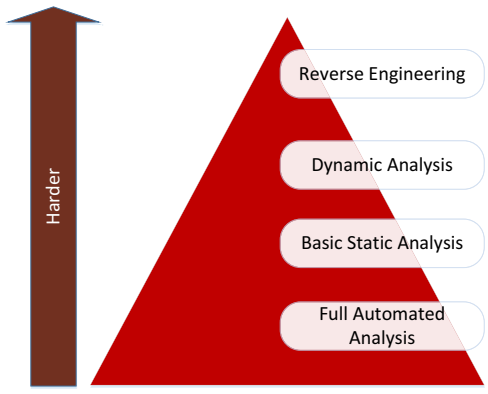


Figure 2. Stages of present-day Malware analysis

### B. *Basic Static Malware Analysis*

Static Malware analysis is when the malware is examined without being executed for example string analysis [8]. Many automated static analysis tools are available and

although we include them in the full automated analysis stage, their main purpose is to perform static analysis. The tools use known syntax or structural properties of the malware code to extract information from the file. Sometimes, an analyst uses the conventional command line based analysis to extract information. The information collected during this type of analysis is very simple and not always sufficient for a conclusive decision on the malicious intent of the file. It is however advised to start with this type as it can provide information to provide the direction for the next analysis phase for a specific malware. However, analysis evasion techniques like packing and encryption normally lead to this analysis step providing incorrect information or information that is not useful.

### C. Dynamic Malware Analysis

Dynamic Malware analysis involves analysing the program while it is running on a system [22]. The malware is run in a "safe" and controlled environment to avoid transference of the malware to other systems or networks.

Basic dynamic malware analysis involves observing collected sample interacting with a system. A snapshot of the original state of the virtual machine is taken, the malware is then introduced into the system and executed. The new state and the original state are compared for changes. The observed changes are then used to remove the infection from infected systems and/or modelling effective signatures. Like basic static malware analysis, it is an important initial step of malware analysis though it does not provide exhaustive information on the malware [8], [23].

Advanced dynamic analysis involves using tools to examine the state of the executed malware as it is running. The internal state of the malicious code is examined and this technique provides information that would normally be impossible to gather when using other techniques [8]. The analysis is always run in a controlled environment to ensure that all the inputs and output of the system are known and their effects can be accounted for. Various tools used at this stage will monitor the APIs, system function calls invoked, files created and or deleted, registry changes and the data processed by the program under analysis as it interacts with the system. Analysis of the parameters used during the API and function calls allows for the functions used to be grouped semantically while analysis of the data processed and propagated within the system gives an understanding of the files used and produced by the malware. These lead to the identification of the tasks that the malware undertakes to fulfil its functionality [24]. Advanced dynamic malware analysis is very helpful in identification of malware variants and obfuscated techniques.

Automated dynamic malware analysis tools give reports that can be used to group the malware according to behaviour [24] and have been explored in the fully automated analysis section.

### D. Reverse Engineering

Reverse engineering, also called advanced static malware analysis involves loading the executable into a disassembler to loop through its execution process and then examining the instructions to understand what the program does [14], [23]. Debugging tools like IDA Pro, OllyDbg and WinDbg are few of the tools that are normally used at this stage. In advanced static analysis/ reverse engineering, analysis evasion techniques like instruction replacement can produce ambiguous results and malware that needs input information that cannot be statistically determined for example time and date can be hard to analyse [8], [25].

There is no defined specific analysis order that an analyst follows to obtain the needed information [18]. It is normally up to experience and skill.

## V. Malware Analysis and Detection Evasion.

The first known malware to exhibit detection avoidance was the Brain virus, written in 1986 by the Farooq Alvi brothers [5]. Any attempt to read the virus infected disk section led to the computer displaying clean data instead of the infected lot. Since then, advancements in discovered malware all show that the survival is the number one priority since the longer the malware is undetectable, the more profitable it is to the writer [26].

For malware to survive in any system, it must avoid being detected by the many security measures employed which include anti-virus software, firewall, and intrusion detection systems among others. Therefore, the evolution of these techniques adds another dimension to the challenge of the fight against malware [5]. Some of the known malware analysis and detection evasion techniques mainly target avoiding being fully analysed or being detected while other serve to evade both analysis and detection.

Originally, anti-reverse engineering was used in legitimate software as a way of protecting intellectual property by software companies and individuals [25]. Malware writers then adopted the idea of anti-analysis, the tools and also created customised versions to evade detection [8]. In this section, we explore some of the known malware analysis and detection evasion techniques which we have classified into 2 categories; Anti-Analysis and Obfuscation techniques. We also later describe the various types of malware variants known.

### A. Anti-Analysis Techniques:

These are tricks used by malware to hinder the analysis process [27]. If these tricks go undetected, they can lead to dead ends, wrong information or being stuck in an infinite execution loop.

#### 1) Anti- File Processing

In file parsing also called file processing malware detectors use the known file structure to parse the file contents as shown in Figure 1. This allows for detection of malware using heuristic based analysis where extracted file features are used to determine if the file is malicious or not.

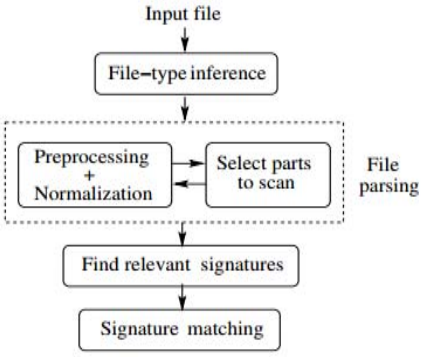


Figure 1. File Processing in Anti-virus systems[28]

Malware employ anti-file processing method to thwart scanners that rely on known file formats leading to the file parsers throwing errors instead of extracting the file features. There are two identified exploits in this technique [28]:

- Chameleon attacks: It leads to the file appearing as one file type to the file parser while appearing as a different type to the target operating system.
- Werewolf attacks: This creates malformations in the file format that leads to the file parser being unable to extract information.

*2) Anti- Debugging*

Anti-debugging techniques can be deployed by hooking to various interrupts, using interrupts to generate new decryption keys, through the use of runtime code checksums, checking debugging API routines loaded, checking various registry keys (according to a particular debugger software), using registers and stacks [8]. Some malicious programs are able to detect the presence of debuggers and then give the wrong output or unexpected events [29].

*3) Anti- Virtual Machine*

Running malware in a virtual machine is common and considered a safe method used to analyse the behaviour of the malware as it theoretically infects the virtual machine and never the host [14]. Virtualisation or malware sandboxing also enables faster analysis times than installing new hosts every time one needs to examine a new malware sample. To thwart the analysis of samples inside a virtual machine, malware include anti-VM protection or they simply exit when the malware detects that is running in an isolated environment.

Anti-VM techniques can be deployed by detecting whether they are running in a virtual machine or not. This can be achieved either by looking at VME artefacts in processes, file system, registry and memory or by looking for VME-specific virtual hardware, processor instructions and capabilities [30].

## *B. Obfuscation Techniques*

Sometimes, once a program (P) is written, the program lines can be re-ordered or lines can be replaced without affecting the intended behaviour resulting into an equivalent but transformed program (P') [8]. Obfuscated malware perform the same function as the original malware only that the signature is changed. Obfuscation techniques serve as anti-disassembly techniques since they try to thwart the reverse engineering process once the sample is loaded into a debugger. These are some of the techniques identified under malware obfuscation:

*1) Binary Obfuscation*

Binary obfuscation techniques are methods of detection avoidance that are applied after the program has been compiled.

*Encryption*

Malware encrypt the original code or blocks of the original code transforming the code into blocks of code that do not make sense to the human eye. Malware writers are known to use existing encryption techniques. Since more encryption techniques like the base64 encoding and ROT operators can be decrypted once the pattern is realised, malware writers complicate their patterns by customising their patterns for example creating their own alphabets [31].

*Entry point obfuscation*

The Entry point obfuscation (EPO) techniques is implement by the malware scanning the host file and then changing the pattern of certain API especially the ExitProcess to point to the beginning of the malicious execution code[30], [32]. This way, the program relies on a function call to get executed and not on the operating system loader. The malicious code is embedded safely inside the host file at a random location which this then called by the function call. Once it is executed, it passes the call routine control back to the actual subroutine.

*Packing*

The most common and default feature of packers is compressing the file into a smaller size. The packer works as an envelope that hides the program from any outside sources until it is run in the system. It also pre-pends the unpacker to the newly formed program which is the encrypted original program stored as data in the new executable [8]. This was initially developed by commercial software companies when disk size was of prime importance [33]. However, malware writers have adopted and misused it. In the analysis of malware especially when trying to reverse engineer the program, a packed program is easy to detect using analysis tools. With the introduction of custom built packers, reverse engineers manually unpack the code using debuggers because of lack of the original packer information.

*Stealth Methods*

A stealth malware is a type of malware that tries to remain undiscovered by hiding the infection events [5], instead of trying to obfuscate its code. It achieves this by restoring certain original properties of the host file for example, timestamps. It also intercepts system calls to hide any other resulting changes like the increase in the size of the host file. Other techniques used are creating alternate data streams (NTFS) [31] to infect files with malware.

*Condition based Execution*

Some Malware require specific inputs and unless these inputs are met, they do not execute or provide wrong results. Conditions like specific days or presence of specific features on the system. Unless these inputs are known previously and emulated, analysis of this malware not possible [14].

*2) Code Obfuscation*

Code Obfuscation techniques are applied to the program during the writing of the code itself. These methods are employed to change the syntax of the program without changing the semantics of the program. Some of the known methods are [34]:

- Dead-Code Insertion; where "trash-code" lines are added in the program without changing the behaviour of the code.
- Code Transposition where the code is shuffled so that the binary order is different from the execution order.
- Register Reassignment which uses the replacement of registers with others within a specific code live range.
- Instruction Substitution where equivalent instructions are used to replace existing ones.
- Code Integration where the virus code is interweaved into the code of its target program.

### C. *Types of Obfuscated Malware*

Obfuscated malware are defined by the structural and syntactic similarities and differences to already existing malware. These are grouped into 3 groups; packed, metamorphic and Polymorphic malware depending on the detection evasion technique used.

#### 1) *Packed Malware*

Most malware writers apply packers or even multiple packers to produce different variants of the same malware code. Perdisci, et al [33] states that more than 80% of the new malware discovered are actually packed versions of already existing malware. Packers compress the file into a smaller size and sometimes encryption is applied to the compressed version of the file to make the unpacking process more difficult. Some packers are custom built by malware writers and these can be used to actually detect that the file is malicious without the need for further analysis while commercial packers that are readily available online are seen in many malware variants [31].

#### 2) *Oligomorphic Malware*

Also called 'Semipolymorphic'[35] malware, they employ multiple decryption routines which are chosen randomly at infection as a way of avoiding signature based detection. The Whale virus was the first malware to use this technique and it carried a few dozens of different descriptors and picked one randomly [36].

#### 3) *Polymorphic Malware.*

Polymorphic malware like oligomorphic malware, use decryption routines to change the look of the execution codes for every infection. They have a wide range of decryption engines since they tend to use mutation engines. The mutation engines perform all the logic computations in rearranging the code to prevent detection by signature matching. The decryptor is run first once the malware is copied to the machine and it enables the execution of the malware. When the malware replicates itself, it encrypts the new malware with a different key and encloses the new decryption routine in the new code. It can however only generate up to a few hundred decryptors so it can be detected [37].

#### 4) *Metamorphic Malware*

The malicious code body is changed instead of appearance by using a combination of various obfuscation techniques. By using dead-code insertion, register reassignment and code transposition, the body of the code is changed into a new generation but it works the same way [38]. This way, every generated variation of the malware looks different and therefore signature generation and signature based detection is very hard. Unlike most polymorphic malware which decrypt to a single constant code body in memory, metamorphic can have varying codes which makes their detection in memory rely on algorithmic scanning [5]. Metamorphic malware can also insert and interweave their code into the host program which makes the malware harder to detect.

## VI. EXISTING MALWARE DETECTION TECHNIQUES AND SOLUTIONS

Malware detection is the process of identifying malicious code from benign code so that the system can be protected or recovered from any effects that the malicious code.

### A. *Integrity Checker*

Compromising a computing system or network requires that some changes be made within the target environment. Integrity checkers are used in intrusion detection on the premise that a file which exists within the uncompromised operating environment is used as a measure to counter any future changes [5]. A hashing function like the md5 sum, Sha1or Sha256 is used to calculate the digest of the program or file and stored in the database. Later, the digests of the program/ file are re-calculated and then compared against the originally calculated hash to check if the file has been modified. The challenges that this method presents are:

- The system initially used to calculate the stored hashes must be deemed clean which is hard to guarantee.
- System updates and patches that very prevalent in computer systems do modify system files and programs which means that the database of hashes needs to be updated for every update or there will very high false positives affecting the detection method.
- There is need to ensure that the reference database of hashes is stored offline and safely otherwise it presents a single point of failure in the detection mechanism.

Integrity checking is considered quite important in detection of any system modifications. It is however more of an incident recovery process method than malware infection prevention method.

### B. *Signature Based Detection*

Signature based detection uses specific byte code sequences that are identified to be unique to a sample of malware in a specific family or variant them to detect the presence of similar coded files in a system [5]. The unique byte of code sequence are saved in the anti-virus database as signatures and are developed by a group of malware experts after detailed analysis a significant number of malware [3]. Any file being scanned by the antivirus that is found to contain the signature of the unique byte code sequence is deemed to be malicious. This implies that a database of

signatures must be maintained by the antivirus and updated every time new signatures are generated in order detect new malware. This creates one of the major challenges faced by system users as updating these signatures requires access to networking resources that might not be readily available all the time.

### C. Semantic Based Detection.

Semantics-based malware detection looks to identify the malware by deducing the logic of the code and matching it to already known malicious logic patterns. It follows the semantics of the code instructions within the file instead of looking at the syntactic properties unlike signature based detection. This allows for the semantic based detection approaches to overcome obfuscation and detect unknown malware variants. An overview of semantic based detection approaches is provided in [39].

### D. Behavioural Based Detection

Behaviour-based detection techniques focus on using specific system/ application behaviour and activities that are observed during dynamic analysis of the sample to form patterns that can be used to identify software that invoke similar patterns. Although these techniques are largely immune to obfuscation, their applicability is limited by their performance as dynamic analysis requires time and determining the unsafe activities and behaviour within the environment is an evolving challenge [40]. An survey of behavioural based malware detection is provided in [41].

### E. Heuristic Based Detection

Heuristic in computing is defined as "Proceeding to a solution by trial and error or by rules that are only loosely defined" [42]. The main idea behind heuristic based detection is that there is no need to know so much about the inner structure or logic of the program being scanned and the main aim to reach as close to a conclusive decision as possible using the best optimal path. Therefore, heuristic based detection approaches use algorithms and or rules that scan for known patterns. The first heuristic approaches are known to have been built in 1989 to detect DOS viruses [43]. Most antivirus programs today use a combination of heuristic engines and signature based scanners. Recent research in the use of datamining for malware detection are considered heuristic based detection approaches [44].

## VII. Conclusion and Future work

This paper reviews the foundational information of malware and anti-malware systems. We presented summaries of works found in literature about malware evolution, malware analysis techniques, malware evasion techniques and existing malware detection methods. We then review literature of recent research approaches to malware analysis and detection. This work was done as foundation for malware detection frameworks that have been developed.


[1] S. Morgan, "Cybercrime damages expected to cost the world $6 trillion by 2021," CSO Online, 22-Aug-2016. [Online]. Available: http://www.csoonline.com/article/3110467/security/cybercrime-damages-expected-to-cost-the-world-6-trillion-by-2021.html. [Accessed: 31-Aug-2016].

[2] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Proactive Detection of Computer Worms Using Model Checking," IEEE Trans. Dependable Secure Comput., vol. 7, no. 4, pp. 424–438, Oct. 2010.

[3] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue Univ., p. 48, 2007.

[4] K. Mathur and S. Hiranwal, "A Survey on Techniques in Detection and Analyzing Malware Executables," vol. Volume 3, Issue 4, April 2013, pp. 423–428, 2013.

[5] E. Skoudis and L. Zeltser, Malware: Fighting Malicious Code. Prentice Hall Professional, 2004.

[6] Bitdefender, "History Of Malware," 2010 2008.

[7] J. Koret and E. Bachaalany, The antivirus hacker's handbook. Indianapolis, IN: John Wiley & Sons Inc, 2015.

[8] M. Sikorski and A. Honig, "Practical Malware Analysis," Netw. Secur., vol. 2012, no. 12, p. 4, 12.

[9] J. Aycock, Spyware and Adware, vol. 50. Boston, MA: Springer US, 2011.

[10] F. Thomas, Adware: The Only Book You'll Ever Need. Lulu Press, Inc, 2015.

[11] C. C. Elisan, Malware, Rootkits & Botnets A Beginner's Guide. McGraw Hill Professional, 2012.

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

[13] C. C. Elisan, Malware, Rootkits & Botnets A Beginner's Guide. McGraw Hill Professional, 2012.

[14] M. W. Ligh, Malware analyst's cookbook and dvd, 1st ed. Indianapolis, IN: Wiley Pub., Inc, 2010.

[15] V. Harrison and J. Pagliery, "Nearly 1 million new malware threats released every day," CNNMoney, 14-Apr-2015. [Online]. Available: http://money.cnn.com/2015/04/14/technology/security/cyber-attack-hacks-security/index.html. [Accessed: 03-Jun-2015].

[16] "Securelist - English - Global - securelist.com," Securelist - Kaspersky Lab's cyberthreat research and reports. .

[17] A.-T. GmbH, "AV-TEST – The Independent IT-Security Institute." [Online]. Available: https://www.av-test.org/en/statistics/malware/. [Accessed: 19-Jun-2017].

[18] "Mastering 4 Stages of Malware Analysis." [Online]. Available: https://zeltser.com/mastering-4-stages-of-malware-analysis/. [Accessed: 27-Oct-2016].

[19] "joxeankoret/pyew," GitHub. [Online]. Available: https://github.com/joxeankoret/pyew. [Accessed: 31-May-2015].

[20] "Malwr - Malware Analysis by Cuckoo Sandbox." [Online]. Available: https://malwr.com/. [Accessed: 27-Oct-2016].

[21] "VirusTotal - Free Online Virus, Malware and URL Scanner." [Online]. Available: https://www.virustotal.com/. [Accessed: 27-Oct-2016].

[22] C. H. Malin, E. Casey, and J. M. Aquilina, Malware Forensics: Investigating and Analyzing Malicious Code. Syngress, 2008.

[23] E. Eilam and E. J. Chikofsky, Reversing: secrets of reverse engineering. Indianapolis, IN: Wiley, 2005.

[24] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," ACM Comput. Surv. CSUR, vol. 44, no. 2, p. 6, 2012.

[25] J. Koret and E. Bachaalany, The Antivirus Hacker's Handbook. John Wiley & Sons, 2015.

[26] "A Brief History of Malware Obfuscation: Part 1 of 2," blogs@Cisco - Cisco Blogs. [Online]. Available: http://blogs.cisco.com/security/a_brief_history_of_malware_obfuscation_part_1_of_2. [Accessed: 27-Oct-2016].

[27] "Five Anti-Analysis Tricks That Sometimes Fool Analysts," Malwarebytes Unpacked. [Online]. Available: https://blog.malwarebytes.org/intelligence/2014/09/five-anti-

debugging-tricks-that-sometimes-fool-analysts/. [Accessed: 03-Jun-2015].

[28] Jana Suman and Vitaly Shmatikov, "Abusing File Processing in Malware Detectors for Fun and Profit." [Online]. Available: https://www.cs.cornell.edu/~shmat/shmat_oak12av.pdf. [Accessed: 10-Nov-2016].

[29] "Anti-debugging and Anti-VM techniques and anti-emulation." [Online]. Available: http://resources.infosecinstitute.com/anti-debugging-and-anti-vm-techniques-and-anti-emulation/. [Accessed: 11-Nov-2016].

[30] J. K. Bachaalany Elias, The Antivirus Hacker's Handbook. .

[31] "Obfuscation: Malware's best friend," Malwarebytes Labs. [Online]. Available: https://blog.malwarebytes.com/threat-analysis/2013/03/obfuscation-malwares-best-friend/. [Accessed: 24-Nov-2016].

[32] M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, Malware Detection. Springer Science & Business Media, 2007.

[33] R. Perdisci, A. Lanzi, and W. Lee, "Classification of packed executables for accurate computer virus detection," Pattern Recognit. Lett., vol. 29, no. 14, pp. 1941–1946, Oct. 2008.

[34] C. K. Behera and D. L. Bhaskari, "Different Obfuscation Techniques for Code Protection," Procedia Comput. Sci., vol. 70, pp. 757–763, Jan. 2015.

[35] J. Aycock, Computer Viruses and Malware. Springer, 2006.

[36] P. Szor, The Art of Computer Virus Research and Defense. Pearson Education, 2005.

[37] Y. Ilsun and Y. Kangbin, "Malware Obfuscation Techniques: A Brief Survey," presented at the Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, 2010, pp. 297–300.

[38] P. O'Kane, S. Sezer, and K. McLaughlin, "Obfuscation: The Hidden Malware," Secur. Priv. IEEE, vol. 9, no. 5, pp. 41–47, 2011.

[39] M. D. Preda, M. Christodorescu, S. Jha, and S. Debray, "A Semantics-based Approach to Malware Detection," in Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New York, NY, USA, 2007, pp. 377–388.

[40] R. Luh, S. Marschalek, M. Kaiser, H. Janicke, and S. Schrittwieser, "Semantics-aware detection of targeted attacks: a survey," J. Comput. Virol. Hacking Tech., pp. 1–39, May 2016.

[41] G. Jacob, H. Debar, and E. Filiol, "Behavioral detection of malware: from a survey towards an established taxonomy," J. Comput. Virol., vol. 4, no. 3, pp. 251–266, Aug. 2008.

[42] "heuristic - definition of heuristic in English | Oxford Dictionaries," Oxford Dictionaries | English. [Online]. Available: https://en.oxforddictionaries.com/definition/heuristic. [Accessed: 28-Nov-2016].

[43] "Heuristic Techniques in AV Solutions: An Overview | Symantec Connect." [Online]. Available: https://www.symantec.com/connect/articles/heuristic-techniques-av-solutions-overview. [Accessed: 28-Nov-2016].

[44] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in 2001 IEEE Symposium on Security and Privacy, 2001. S P 2001. Proceedings, 2001, pp. 38–49.