Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 11, November - 2017

Review of Malware and Techniques for Combating Zero Day Attacks

Emmah, Victor Thomas
Department of Computer Science,
Rivers State University, Nigeria

Ejiofor, C. I
Department of Computer Science,
University of Port Harcourt, Nigeria.

Onyejegbu, Laeticia N.
Department of Computer Science,
University of Port Harcourt, Nigeria.

***Abstract*- Zero-day attacks have become a very important security issue that should not be overlooked. Malwares are used to infect computer systems thereby causing malicious intent. The frightening issue is that software vendors are not aware of the vulnerability available in the software and as such no warning is given before the attackers strike. This paper discusses the different types of malwares that are used for attacks as well as some of the techniques for combating zero-day attacks. Finally, machine learning techniques for efficient and effective detection of malware are also discussed.**

Keywords -Machine learning, Malware, Zero-day attacks.

1. BACKGROUND OF STUDY

Today, the internet has become a persistent threat environment for various types of organisations. The proliferation of newly developed technologies which are being adopted by different organisations for their changing business needs, are taken advantage of by malicious or sneaky sources that lie in wait to exploit vulnerabilities in them. Zero-day attacks have dominated the headlines over the years for political, social and monetary gains. According to Symantec's Internet Security Threat Report of 2014, there is 91% increase in targeted attacks campaigns in 2013, 62% increase in the number of security breaches and 23 zero-day vulnerabilities were discovered [23]. Large tech companies like Apple, Facebook, Microsoft, Twitter and others are also being targeted with same zero-day Java vulnerability that attacks multiple customers [24].
These facts and figures show that there is a very serious concern in today’s network security; and the zero day attacks are among the top security concerns that the modern enterprises face. The reality today is that every industry and organisation faces zero-day attack. Every day, companies loose sensitive data because of various security breaches they encounter.

Malware is a software or computer program used to perform these malicious actions. The term is a combination of the words ‘MALicious’ and ‘softWARE’. The end goal of most cyber criminals is to install malware on your computers or mobile devices. Once installed, these attackers can potentially gain total control of them. Malware has been misconceived to be a problem only for Windows computers. While Windows is widely used, and thus a big target, malware can infect any computing device, including smartphones and tablets. In fact, the prevalence of malicious software infecting mobile devices is steadily growing [28].

Malware is no longer created by just curious hobbyists or amateur hackers, but by sophisticated cyber criminals to help them achieve specific goals. These goals can include stealing confidential data, harvesting logins and passwords, sending spam emails, launching denial of service attacks, extortion or identity theft. For example, malware known as Cryptolocker is used by cyber criminals to infect and encrypt all of the files on your computer. Once infected and encrypted, these cyber criminals then demand a ransom in exchange for decrypting your files [28].

2. TYPES OF MALWARE

There are different types of malicious software which we will discuss in this section. Each of them gain entrance into the computer system and attack the system in different ways.

Figure 1. Different types of malware

2.1. Trojans

A Trojan is a hidden threat, much like the famed Trojan horse left by Odysseus on the shores of Troy. Simply put, a Trojan consists of two parts—a server side that runs on an attacked host and a client piece that runs on the attacker’s console. The server code (usually kept very small in size, no more than a few KBs) is dispatched to the victim via some malware distribution method. In a simple setting, the attacker sends the victim a file that contains the server code (e.g. an image or a PDF large enough in size that the server size is miniscule when compared to the overall file size). When the user double-clicks the attacked file, it launches the “server” program embedded in the infected file. The server usually runs in stealth mode and is not easily visible to the user and/or to the file manager [22].

At this stage, the server code in the infected file can establish contact with the attacker’s client code in one of many ways. One simple way is through a reverse connection in which the server code has the IP address from which the attacker wants to control the victim’s computer. But much more sophisticated reverse connection methods also exist. Once launched, the server program contacts the client side code from whose console, the attacker can now take control of the victim’s program. He can install new programs on to the victim’s computer (e.g. keyloggers), he can read every single file on the victim's computer (e.g. credit card and banking information, personal identity information), and more. In effect, the victim's computer can be controlled from a remote location.

An interesting Trojan, Obad.a infects Android devices by first sending potential victims an infected link (or a spam message). When the victim clicks the link, he downloads the Trojan server onto his device which immediately reaches out to his entire contact list, urging them to click on the link as well. The Trojan spreads in this way, infecting a large number of people. Unlike most Trojans, this one uses a botnet to control the spread of the Trojan [24].

2.2. Worms

A worm is a piece of malware that can independently spread through a network by exploiting vulnerabilities in existing software to compromise a system [27]. Worms may spread through networks in a variety of ways. For instance, worms may spread through a network by using email to infect other computers, or by using other file transfer protocols to copy themselves onto other computers. Worms may carry a payload. While some worms may do nothing other than spread from one computer to another (just using up bandwidth and slowing down a network), others may do dangerous things like delete files on a machine and encrypt files (so that the owner of the file has to pay a ransom in order to be able to decrypt his files).
Some of the notorious computer worms include stuxnet, conficker, Mydoom etc.

2.3. Viruses

Unlike worms, that spread independently, viruses spread by attaching themselves to another program or to files (e.g. PDF or image files). For example, a virus embedded in a PDF or JPEG file may spread when that file is opened [22]. Some viruses also exist in the boot sector of a computer hard drive, thus executing automatically when a boot operation takes place. Because legitimate programs and files have well known sizes, viruses that attach themselves to such "entities" may take steps to hide any increase in size. One way to hide is by copying themselves into unused space in a file or program. Another way to hide is by intercepting requests to obtain data about the program or file and returning results that appear normal and obfuscate the presence of the virus. In order to hide from "signature based" scanners used by many anti-virus companies (a signature is just a fragment of code), viruses can mutate, making their code look different. Rates of mutation vary from one virus to another [22].
It is unfortunate that in common parlance, the word "virus" has been collectively used to describe all kinds of malware including worms, Trojans, and viruses as described above.

2.4. Rootkits

Originally, a *rootkit* was a set of tools installed by a human attacker on a Unix system, allowing the attacker to gain administrator (root) access. Today, the term rootkit is used more generally for concealment routines in a malicious program. Once a malicious program is installed on a system, it is essential that it stays concealed, to avoid detection and disinfection. The same is true when a human attacker breaks into a computer directly [32]. Techniques known as rootkits allow this concealment, by modifying the host's operating system so that the malware is hidden from the user. Rootkits can prevent a malicious process from being visible in the system's list of processes, or keep its files from being read.
In an attempt to keep the user from stopping a malicious process, another is sometimes installed to monitor it. When the process is stopped (killed), another is immediately created. Modern malware starts a number of processes that monitor and restore one another as needed. In the event that a user running Microsoft Windows is infected with such malware (if they wish to manually stop it), they could use Task Manager's 'processes' tab to find the main process (the one that spawned the "resurrector process(es)"), and use the 'end process tree' function, which would kill not only the main process, but the "resurrector(s)" as well, since they were started by the main process. Some malware programs use other techniques, such as naming the infected file similar to a legitimate or trustworthy file (expl0rer.exe VS explorer.exe) to avoid detection in the process list [32].

2.5. Backdoors

A *backdoor* is a method of bypassing normal authentication procedures. Once a system has been compromised (by one of the above methods, or in some other way), one or more backdoors may be installed in order to allow easier access in the future. Backdoors may also be installed prior to malicious software, to allow attackers entry.

2.6. Spyware

Spyware is a type of malicious software that can be installed on computers, and which collects small pieces of information about users without their knowledge. The presence of spyware is typically hidden from the user, and can be difficult to detect. Typically, spyware is secretly installed on the user's personal computer.
While the term spyware suggests software that secretly monitors the user's computing, the functions of spyware extend well beyond simple monitoring. Spyware programs can collect various types of personal information, such as Internet surfing habits and sites that have been visited, but can also interfere with user control of the computer in other ways, such as installing additional software and redirecting Web browser activity. Spyware is known to change computer settings, resulting in slow connection speeds, different home pages, and/or loss of Internet connection or functionality of other programs. In an attempt to increase the understanding of spyware, a more formal classification of its included software types is provided by the term *privacy-invasive software.* [32].

Classification of code as spyware (or sometimes browser cookies as "tracking" cookies) can be controversial. Often the software is installed by the user knowing that some amount of monitoring will take place. (Users generally agree to this activity to get free software and it is often associated with music and video sharing.) Some such software allows the user to turn off the monitoring, assuming they are aware of it and can find instructions for disabling it. Anti-spyware is usually part of anti-virus programs; scan using at least two different AV packages. Spybot Search and Destroy is a good freeware program for looking for spyware (but it is not an Anti-Virus program) [32].

2.7. Loggers

Keystroke logging (often called keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored. It is a hardware device or a software program that records the real time activity of a computer user including the keyboard keys they press. A keylogger, when installed, can generally do the following [14]:

- capture any passwords entered by users on the device
- take screen captures of the device at periodic intervals
- record the URLs that were visited via Web browsers, and possibly also take screen captures of the Web pages viewed
- record a list of the applications run by users on the device

- capture logs of all instant messaging (IM) sessions
- capture copies of sent emails
- automatically send the reports containing stored logs and emails to a remote location (by email, FTP or HTTP).

There are numerous keylogging methods, ranging from hardware and software-based approaches to electromagnetic and acoustic analysis. Key logging is often used by law enforcement, parents, and jealous or suspicious spouses (lovers). The most common use, however, is in the workplace, where your employer is monitoring your use of the computer. Unfortunately, all of these activities are legal [32].

2.8. *Adware*

Adware, or advertising-supported software, is any software package which automatically plays, displays, or downloads advertisements to a computer. These advertisements can be in the form of a pop-up. The goal of the Adware is to generate revenue for its author. Adware, by itself, is harmless; however, some adware may come with integrated spyware such as keyloggers and other privacy-invasive software. Advertising functions are integrated into or bundled with the software, which is often designed to note what Internet sites the user visits and to present advertising pertinent to the types of goods or services featured there. Adware is usually seen by the developer as a way to recover development costs, and in some cases it may allow the software to be provided to the user free of charge or at a reduced price. The income derived from presenting advertisements to the user may allow or motivate the developer to continue to develop, maintain and upgrade the software product. Conversely, the advertisements may be seen by the user as interruptions or annoyances, or as distractions from the task at hand.

Some adware is also shareware, and so the word may be used as a term of distinction to differentiate between types of shareware software. What differentiates adware from other shareware is that it is primarily advertising-supported, like many free smartphone apps. Users may also be given the option to pay for a "registered" or "licensed" copy to do away with the advertisements. Pandora Radio offers both a free version (with ads) and a paid subscription (without ads) [32].

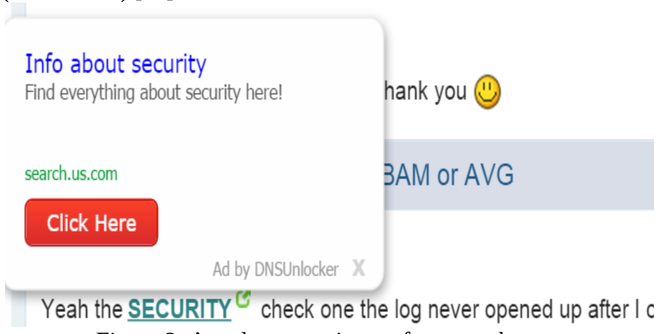
Info about security
Find everything about security here!
search.us.com
Click Here
Ad by DNSUnlocker X
hank you
BAM or AVG
Yeah the SECURITY check one the log never opened up after I c

Figure 2. An adware poping up from a webpage

There is a group of software (Alexa toolbar, Google toolbar, Eclipse data usage collector, etc.) that send data to a central server about which pages have been visited or which features of the software have been used. However differently from "classic" malware these tools document activities and only send data with the user's approval. The user may opt in to share the data in exchange to the additional features and services, or (in case of Eclipse) as the form of voluntary support for the project. Some security tools report such loggers as malware while others do not. The status of the group is questionable. Some tools like PDF Creator are more on the boundary than others because opting out has been made more complex than it could be (during the installation, the user needs to uncheck two check boxes rather than one). However, PDF Creator is only sometimes mentioned as malware and is still subject of discussions [32].

These malwares can however be distributed through different methods in order to harm the system or device and thus cause zero-day attacks. The distribution methods can be Drive-by-Downloads which involves the user unknowingly downloading a malicious file from the internet or emails, network intrusion where attackers run programs on victim's computers and manipulate data packages to exploit vulnerabilities, or through social engineering techniques where attackers exploit human weaknesses by manipulating users into running malicious binaries [18][22].

3. ANALYZING AND COMBATING MALWARE

An integral component in security breaches are malicious software or malware; therefore analysing malware by dissecting it to understand how it works, how to identify it, and how to defeat and eliminate it is important [20]. Malware analysis is a critical task for responding to computer or network security incidents as it allows better assessments of the nature of a security incident and may even help to prevent further infections [9]. Malware analysis can also be used to develop host-based signatures to identify files or registry keys on a victim computer that indicate an infection; or network-based signatures by analysing network traffic [5]. The goal of malware analysis is to gain an understanding of how a specific piece of malware functions so as to build defences to protect an organisation's network [10]. Malware analysis can either be done statically or dynamically.

3.1. *Static Malware Analysis:*

In static analysis, the capabilities of the malware are learned by examining the code from which the program is compromised. Static or Code Analysis is usually performed by dissecting the different resources of the binary file *without executing it* and studying each component [29]. The binary file can also be disassembled (or reverse engineered) using a disassembler such as IDA. If the source code is available, information such as variables, data structures, used functions and call graphs can be extracted. The machine code can sometimes be translated into assembly code which can be read and understood by humans: the malware analyst can then make sense of the assembly instructions and have an image of what the program is supposed to perform. Some modern malware is authored using evasive techniques to defeat this type of analysis, for example by embedding syntactic code errors that will confuse disassemblers but that will still function during actual execution [20].
Static analysis can therefore include the following techniques:

i. File Format Inspection: File metadata can provide useful information. For example, Windows PE (portable executable) files can provide much information on compile time, imported and exported functions, etc.
ii. String Extraction: This refers to the examination of the software output (e.g. status or error messages) and inferring information about the malware operation.
iii. Fingerprinting: This includes cryptographic hash computation, finding the environmental artifacts, such as hardcoded username, filename, registry strings.
iv. AV scanning: If the inspected file is a well-known malware, most likely all anti-virus scanners will be able to detect it. Although it might seem irrelevant, this way of detection is often used by AV vendors or sandboxes to "confirm" their results.
v. Disassembly: This refers to reversing the machine code to assembly language and inferring the software logic and intentions. This is the most common and reliable method of static analysis.

The main advantage of static analysis is the ability to discover all possible behavioral scenarios. Researching the code itself allows the researcher to see all ways of malware execution that are not limited to the current situation. This type of analysis is safer, since the file is not executed and it cannot result in bad consequences for the system. Because of these reasons it is not usually used in real-world dynamic environments, such as anti-virus systems, but is often used for research purposes, e.g. when developing signatures for zero-day malware [17].
Static analysis also offers a significant improvement in malware detection accuracy; but its main weakness lies in the difficulty to handle obfuscated and self-modifying code and it is time-consuming. This problem is addressed with the use of a dynamic analysis approach.

3.2. Dynamic Malware Analysis:

Dynamic or Behavioral analysis is performed by observing the behavior of the malware while it is actually running on a host system. This form of analysis is often performed in a sandbox environment to prevent the malware from actually infecting production systems; many such sandboxes are virtual systems that can easily be rolled back to a clean state after the analysis is complete [29]. The malware may also be debugged while running using a *debugger* such as *GDB* or *WinDbg* to watch the behavior and effects on the host system of the malware step by step while its instructions are being processed. Modern malware can exhibit a wide variety of evasive techniques designed to defeat dynamic analysis including testing for virtual environments or active debuggers, delaying execution of malicious payloads, or requiring some form of interactive user input [29].
Generally, there are two main approaches for dynamic malware analysis.

- **i.** ***Analysing the difference between defined states:*** In this approach a given malware is executed for a particular time period and later on, the modifications made to the system are analysed by comparing the current system state to the initial system state. The report from the comparison can be used to identify the behaviour of the malware [9].
- **ii.** ***Observing run-time behaviour:*** This approach uses a specialised tool to monitor the malicious activities initiated by the malware during execution [6].

4. DETECTION TECHNIQUES FOR ZERO-DAY ATTACKS

The research community has proposed several techniques in the defence against zero-day attacks. These detection techniques can be classified as either host-based or network-based. Host-based systems detect the attack at the system level once the attack reaches the vulnerable application and was processed. The network-based systems detect attack at the network level as the attack data travel across the network in the form of packets.

These zero-day attacks can take the form of polymorphic worms, viruses, Trojans, and other malware. According to [8], the most effective attacks that avoid detection are polymorphic worms which show distinct behaviors. “This includes: complex mutation to evade defenses, multi-vulnerability scanning to identify potential targets, targeted exploitation that launches directed attacks against vulnerable hosts, remote shells that open arbitrary ports on compromised hosts to connect to at a later time, malware drops in which malicious code is downloaded from an external source to continue propagation” [8].

The host-based detection installs software on the machine to be monitored and because it runs on the machine itself, the level analysis is deeper compared to the network-based. The network-based detection systems on the other hand, can monitor multiple systems and are able to detect and contain the attacks in their early stage because initially the number of machines compromised is limited. Hence, it is unlikely that a host will see the early attack packets and be able to respond in the early critical period of attack [12].

Most widely deployed intrusion detection systems are network-based due to their simplicity and the ability to operate in real time. The network-based zero-day attack detection techniques that this research focuses on can be broadly classified into: Statistical-based, signature-based, behaviour-based and hybrid-based techniques.

4.1. Statistical-Based Detection Technique

The concept of statistical-based detection is to determine normal network activity and to flag out activities that are not normal or that falls outside its scope. It relies on attack profiles built off of historical data. This means that it relies on attack profiles from past exploits that are now publicly known. From those known exploits, this defence technique adjusts the historical exploit’s profile parameters to detect new attacks. This approach however does not usually adapt well to changes in zero-day exploit data patterns. Any changes in a zero-day exploit’s pattern would require a new profile to be learned by the system [8].

The quality of the detection is directly related to threshold limits set by the vendor or security professional using this technique. Therefore setting the limit (or detection parameters) for judging new observations is a critical step in designing a detection approach since it has a dramatic effect on the quality of the detection. This technique determines what normal activity is and anything outside of normal is blocked or flagged. The longer the system that is utilizing this technique is online, the more accurate the system is at learning or determining what normal is. The detections may frequently result in high rate of false positive or false negatives depending on whether the threshold value is very narrow or wide.

4.2. Signature-Based Detection Technique

Signature-based detection is often used by virus software vendors who will compile a library of different malware signatures. They will cross reference these signatures with local files, network files, email or web downloads depending on settings chosen by the user. These libraries are constantly being updated for new signatures that often represent the signatures of new exploited vulnerabilities. The technique is often one step behind a zero-day exploit because this technique requires a signature to be in the signature library for detection. This is the reason virus software vendors are frequently updating their virus definitions (Hammarberg, 2014).

The signature-based detection techniques mainly focus on zero-day polymorphic worms. Polymorphic worm is a type of worm which changes itself after each step of its propagation keeping its semantics intact. Like other worms, they have the characteristic that it has some invariant byte stream but the sequence of these bytes is highly random. With every exploit, the worms tend to change the byte stream by removing some code portion, inserting some byte sequence or modifying certain bytes. This characteristic of polymorphic worms poses a great challenge to the security professionals leading to the development of new systems. Several polymorphic worm signature generation schemes are surveyed. Based on their characteristics, the signatures are classified into four categories namely: content-based, flexible content-based, semantic-based and vulnerability-driven signatures [9].

4.3. Behaviour-Based Detection Technique

The activity of a program can be viewed as malicious or benign based on the requirements of the code. "Behavior-based techniques look for the essential characteristics of worms which do not require the examination of payload byte patterns" [8]. The goal of such techniques is to predict the future behaviour of a web server, server or victim machine in order to deny any behaviors that are not expected. Those behaviors are learned by the current and past interactions with the web server, server or victim machine [1]. The technique focuses on the actual dynamics of the malware execution to detect them. It monitors the dynamic behaviour of malicious activity rather than its static characteristics because no matter what, a piece of malware will behave badly while running. This technique relies on the ability to predict the flow of network traffic. Unlike anomaly detection, a program or file is not previously classified as "good" or "bad". The executing processes are monitored to analyse their behaviour in a controlled simulated environment. It is an effective way to detect new threats without waiting for them to any harm. Some of the research works carried out on behaviour-based detection are outlined below.

Network-Level Emulation [19] is a heuristic detection method to scan network traffic streams for the presence of previously unknown polymorphic shellcode. Their approach relies on a NIDS-embedded CPU emulator that executes every potential instruction sequence in the inspected traffic, aiming to identify the execution behavior of polymorphic shellcode. The proposed approach is robust to obfuscation techniques like self- modifications and non-self-contained polymorphic shellcodes.

SGNET [11] is a distributed framework to collect rich information and download malware for zero-day attacks. It automatically generates approximations of the protocol behavior in form of Finite State Machines (FSMs). Whenever the network interaction falls outside the FSM knowledge (newly observed activity), SGNET takes advantage of a real host to continue the network interaction with the attacker. In that case, the honeypot acts as a proxy for the real host. This allows building samples of network conversation for the new activity that are then used to refine the current FSM knowledge.

ENDMal [26] is an anti-obfuscation, scalable and collaborative malware detection system. It consists of multiple monitors where each monitor takes charge of a network area and receives suspicious programs from end-host. Each monitor uses Iterative Sequence Alignment (ISA) method to defeat malware obfuscation and utilizes Handle dependences and Probabilistic Ordering Dependence (HPOD) technology to represent the program behaviors. All the monitors collaboratively identify the malicious program families by sharing HPOD-based behaviors via RENdezvous-based Sharing infrastructure (RENShare), based on Distributed Hash Tables (DHT).

4.4. Hybrid Techniques

This is a combination of heuristics of two or more techniques to form a more sophisticated system for the detection of malwares to fight zero-day attacks. Statistical, signature-based and behaviour-based techniques are combined to form a hybrid system.

[9] proposed a real-time zero-day attack detection and analysis system which combines anomaly-based detection, behavior-based detection and signature-based detection techniques. The proposed system tries to provide a solution to the problem of zero-day attacks. It does so by a layered designed where each layer is dedicated to a single functionality and works in parallel to improve performance.

Honeyfarm [7], a hybrid scheme combines anomaly and signature detection with honeypots. This system takes advantage of existing detection approaches to develop an effective defense against Internet worms. The system works on three levels. At first level, signature based detection is used to filter known worm attacks. At second level, an anomaly detector is set up to detect any deviation from the normal behavior. In the last level, honeypots are deployed to detect zero day attacks. Low interaction honeypots track attacker activities while high interaction honeypots analyze new attacks and vulnerabilities.

[4] presented machine learning methods for malware detection and classification. The purpose was to determine the best feature extraction, feature representation, and classification methods that result in the best accuracy when used on the top of Cuckoo Sandbox. Different classifiers were evaluated with 1156 malware file of 9 families of different types and 984 benign files of various formats. From the author's result, Random Forest method was recommended to implement the classification for multi-class classification, as it resulted in the best accuracy and high performance.

Comar et al (2013) combined supervised and unsupervised learning for Zero-day Malware detection. In their work, they presented a novel machine learning based framework to detect known and newly emerging malware at a high precision using layer 3 and layer 4 network traffic features. Their framework leverages the accuracy of supervised classification in detecting known classes with the adaptability of unsupervised learning in detecting new classes. They proposed an architecture which consist of six major components namely, (i) Data capture, (ii) An intrusion detection/prevention system, (iii) Information storage, (iv) feature extraction and transformation, (v) Supervised classifier, and (vi) a UI portal.

5. MACHINE LEARNING TECHNIQUES

Data mining techniques and methods have developed rapidly thereby giving rise to Machine learning which forms a separate field of Computer Science. Machine learning can be viewed as a subclass of Artificial Intelligence, where the main idea is the ability of a system (computer program, algorithm, etc) to learn from its actions. It was firstly referred to as "field of study that gives computers the ability to learn without being explicitly programmed" by Arthur Samuel in 1959. A more formal definition is given by T. Mitchell in 1997: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E*."
The basic idea of any machine learning task is to train the model, based on some algorithm, to perform a certain task: classification, clusterization, regression, etc. Training is done based on the input dataset, and the model that is built is subsequently used to make predictions. The output of such model depends on the initial task and the implementation.
Two of the most widely adopted machine learning methods are supervised and unsupervised learning

5.1. Supervised Learning

In supervised learning, algorithms are trained using labelled examples, such as an input where the desired output is known. The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Examples of supervised learning are regression and classification problems:

i. Regression
Predict the value based on previous observations, i.e. values of the samples from the training set. Usually, we can say that if the output is a real number and is continuous, then it is a regression problem.

ii. Classification
Based on the set of labeled data, where each label defines a class, that the sample belongs to, we want to predict the class for the previously unknown sample. The set of possible outputs is finite and usually small. Generally, we can say that if the output is a discrete/categorical variable, then it is a classification problem. [4].

5.2. Unsupervised Learning

Unsupervised Learning is used against data that has no historical labels. In contrast to Supervised Learning, in Unsupervised learning, there is no initial labelling of data. Here the goal is to find some pattern in the set of unsorted data, instead of predicting some value. Only the input pattern is given; the network tries on its own to identify similar patterns and to classify them into similar categories A common subclass of unsupervised learning is Clustering

iii. Clustering
Find the hidden patterns in the unlabeled data and separate it into clusters according to similarity. An example can be the discovery of different customer groups inside the customer base of the online shop [4].

6. CLASSIFICATION METHODS

From machine learning perspective, the detection of malware can be done either by classification or clusterization: unknown malware types should be clusterized into several groups, based on certain properties, identified by the algorithm. On the other hand, having trained a model on the wide dataset of malicious and benign files, we can reduce this problem to classification. For known malware families, this problem can be narrowed down to classification only – having a limited set of classes, to one of which malware sample certainly belongs, it is easier to identify the proper class, and the result would be more accurate than with clusterization algorithms [4]. In this section, we give the theoretical background of some of the malware classification methods.

6.1. K-Nearest Neighbours (KNN)

K-Nearest Neighbors (KNN) is one of the simplest, though, accurate machine learning algorithms. KNN is a non-parametric algorithm, meaning that it does not make any assumptions about the data structure. In real world problems, data rarely obeys the general theoretical assumptions, making non-parametric algorithms a good solution for such problems. KNN model representation is as simple as the dataset – there is no learning required, the entire training set is stored.
KNN can be used for both classification and regression problems. In both problems, the prediction is based on the k training instances that are closest to the input instance. In the KNN classification problem, the output would be a class, to which the input instance belongs, predicted by the majority vote of the k closest neighbors. In the regression problem, the output would be the property value, which is generally a mean value of the k nearest neighbours [4]. Figure 3 and Figure 4 shows how the k-Nearest neighbours algorithm works

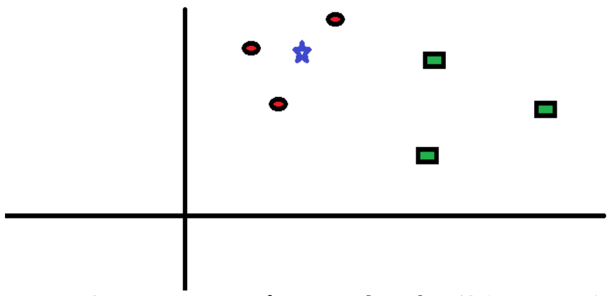

Figure 3. Representation for KNN algorithm (*Srivastava, 2014)*
You intend to find out the class of the blue star (BS). BS can either be Red Circle (RC) or Green Square (GS) and nothing else. The "K" is KNN algorithm is the nearest neighbors we wish to take vote from. Let's say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three datapoints on the plane. Refer to Figure 2.7b diagram for more details:

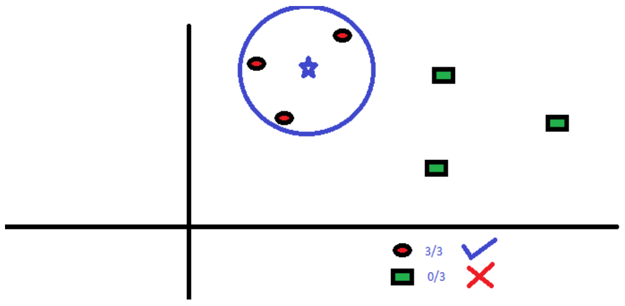

Figure 4. Making a decision with KNN algorithm (*Srivastava, 2014)*

The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm [21]. In classifying objects based on closest training instants in the feature space, the object is classified based on a majority vote of its k nearest neighbours at closest distant from the object.

6.2. Naive Bayes

Naive Bayes is the classification machine learning algorithm that relies on the Bayes Theorem. It can be used for both binary and multi-class classification problems. The main point relies on the idea of treating each feature independently. Naive Bayes method evaluates the probability of each feature independently, regardless of any correlations, and makes the prediction based on the Bayes Theorem. That is why this method is called "naive" – in real-world problems features often have some level of correlation between each other [4].
To understand the algorithm of Naive Bayes, the concepts of class probabilities and conditional probabilities should be introduced first.

1. Class probability is calculated simply as the number of samples in the class divided by the total number of samples:
$$\boldsymbol{P(C)} = \frac{\boldsymbol{count\ (instances\ in\ C)}}{\boldsymbol{count\ (instances\ in\ Ntotal)}} \tag{1}$$
2. Conditional probabilities are calculated as the frequency of each attribute value divided by the frequency of instances of that class.
$$\boldsymbol{P(V|C)} = \frac{\boldsymbol{count\ (instances\ in\ V\ and\ C}}{\boldsymbol{count\ (instances\ in\ V)}} \tag{2}$$

3. Given the probabilities, we can calculate the probability of the instance belonging to a class and therefore make decisions using the Bayes Theorem:

$$\boldsymbol{P}(\boldsymbol{A}|\boldsymbol{B}) = \frac{\boldsymbol{P}(\boldsymbol{B}|\boldsymbol{A})\boldsymbol{P}(\boldsymbol{A})}{\boldsymbol{P}(\boldsymbol{B})} \quad (3)$$

4. Probabilities of the item belonging to all classes are compared and the class with the highest probability if selected as a result.

6.3. J48 Decision Tree

As it implies from the name, decision trees are data structures that have a structure of the tree. The training dataset is used for the creation of the tree, which is subsequently used for making predictions on the test data. In this algorithm, the goal is to achieve the most accurate result with the least number of the decisions that must be made. Decision trees can be used for both classification and regression problems [4]. An example can be seen in Table 1:

Table 1: Decision tree example dataset (Chumachenko, 2017).

Predictions				Target
Outlook	Temperature	Humidity	Windy	Play Tennis
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Overcast	Cool	High	True	No
Rainy	Cool	High	True	Yes
Rainy	Mild	High	False	No
Sunny	Cool	Normal	False	Yes

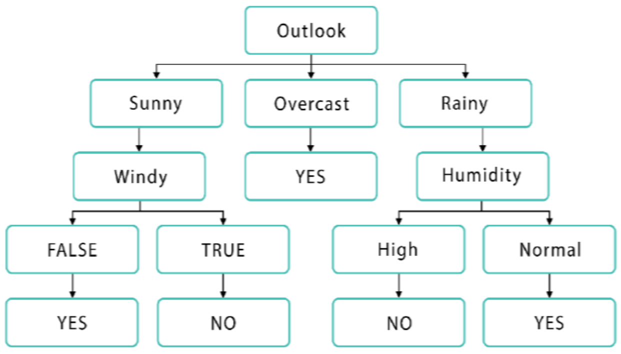

Figure 5. Decision Tree Example (Chumachenko, 2017)

As it can be seen in Figure 2.8, the model was trained based on the dataset and can now classify the tennis playing decision to “yes” or “no”. Here, the tree consists of the decision nodes and leaf nodes. Decision nodes have several branches leading to leaf nodes. Leaf nodes represent the decisions or classifications. The topmost initial node is referred to as root node.

6.4. Random Forest

Random Forest is one of the most popular machine learning algorithms. It requires almost no data preparation and modelling but usually results in accurate results. Random Forests are based on the decision trees described in the previous section. More specifically, Random Forests are the collections of decision trees, producing better prediction accuracy. That is why it is called a ’forest’ – it is basically a set of decision trees. The basic idea is to grow multiple decision trees based on the independent subsets of the dataset. At each node, *n* variables out of the feature set are selected randomly, and the best split on these variables is found.

According to [3], the algorithm can be described as follows:

i. Multiple trees are built roughly on the two third of the training data (62.3%). Data is chosen randomly.

ii. Several predictor variables are randomly selected out of all the predictor variables. Then, the best split on these selected variables is used to split the node. By default, the amount of the selected variables is the square root of the total number of all predictors for classification, and it is constant for all trees.

iii. Using the rest of the data, the misclassification rate is calculated. The total error rate is calculated as the overall out-of-bag error rate.

iv. Each trained tree gives its own classification result, giving its own “vote”. The class that received the most “votes” is chosen as the result.

The scheme of the algorithm is seen in Figure below:

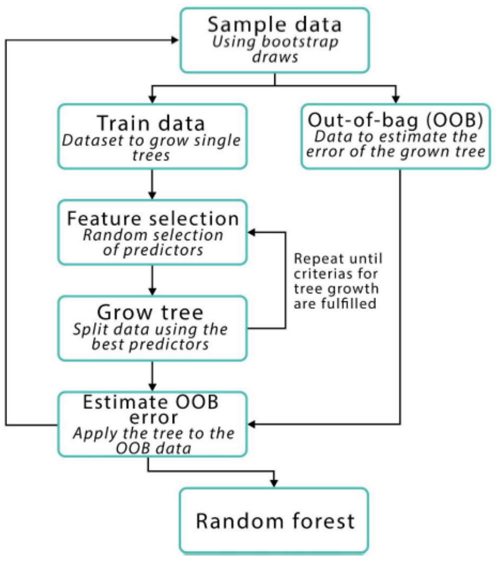
Sample data
Using bootstrap draws
Train data
Dataset to grow single trees
Out-of-bag (OOB)
Data to estimate the error of the grown tree
Feature selection
Random selection of predictors
Repeat until criterias for tree growth are fulfilled
Grow tree
Split data using the best predictors
Estimate OOB error
Apply the tree to the OOB data
Random forest

Figure 6 Random forest Scheme (Chumachenko, 2017)

As in the decision trees, this algorithm removes the need for feature selection for removing irrelevant features – they will not be taken into account in any case. The only need for any feature selection with the random forest algorithms arises when there is a need for dimensionality reduction. Moreover, the out-of-bag error rate, which was mentioned earlier, can be considered the algorithm’s own cross-validation method. This removes the need for tedious cross-validation measures that would have to be taken otherwise. (Mitchell 1997).

Random forests inherit many of the advantages of the decision trees algorithms. They are applicable to both regression and classification problems; they are easy to compute and quick to fit. They also usually result in the better accuracy. However, unlike decision trees, it is not very easy to interpret the results. In decision trees, by examining the resulting tree, we can gain valuable information about which variables are important and how they affect the result. This is not possible with random forests. It can also be described as a more stable algorithm than the decision trees – if we modify the data a little bit, decision trees will change, most likely reducing the accuracy. This will not happen in the random forest algorithms – since it is the combination of many decision trees, the random forest will remain stable [13].

7. CLUSTERING TECHNIQUES

Clustering is a widely used classification mechanism that is designed to categorize data. Such techniques are often used in a data exploration mode, where we are trying to learn something from data that is poorly understood [2]. Cluster Analysis can be done using different algorithms which vary in the methods for determining what belongs to a particular cluster and how proficiently it can find those clusters. Clustering techniques can be performed using raw data, that is, where the class labels for the data are missing and we don't have any information about the data. This can be considered as unsupervised learning. Some clustering techniques are k-means algorithm and Expectation-Maximization clustering.

7.1. K-Means Algorithm

K-means is undoubtedly the most popular clustering technique in use today. Given a set of *m* data points, the algorithm partitions the data into a specified number of mutually exclusive clusters (Alsabti, et.al 1997). In *K*-means, we are given *k*, the number of clusters, and *m* data points *x*1, *x*2, . . . , *xm*. Then we specify one "centroid" for each cluster. Intuitively, a centroid will converge to the center of mass of its cluster. The main goal is to assign a cluster to each of the data point. This method aims to find the means of the clusters such that the distance between the data point to the cluster is minimized. Hence, the problem that we want to solve can be stated as

Given: K and data points $x1$, $x2$, ..., xm

Minimize: $\text{distortion}_k = \sum_{i=1}^{m} d(xi, centroid(xi))$ (4)

where centroid*(xi)* is the centroid to which the data point *xi* belongs. Minimizing the distortion is computationally infeasible, but a simple (and efficient) iterative process often yields a good approximation. K-means algorithm is an iterative process that alternates between two important steps [16]. The two basic steps in this algorithm are as follows:

Step 1: Assign each data point to its nearest cluster.

Step 2: Compute the centroids/means of the cluster based on the points belonging to the cluster. The above two alternating steps form an iterative process that generate a series of solutions that improve as the algorithm proceeds through the iterations.

The algorithm for *K*-means can be given as follows:

1. Given *k*, the data points *x*1, *x*2, . . . , *xm*, and a distance function:
2. Initialize the *k* centroids.
3. For each data point, calculate the distance between that point and each centroid and assign the data point to the cluster corresponding to the nearest centroid.
4. Re-compute the centroids to be the center of each cluster.
5. Goto 3, until there is no (or negligible) change in the clusters [16].

7.2. Expectation –Maximization (EM) Clustering

Expectation-Maximization (EM) clustering is an unsupervised learning technique. EM clustering technique uses Gaussian mixture models and mixture likelihood approach to find a structure in datasets. It follows an iterative approach, which tries to find the parameters of the probability distribution that has the maximum likelihood of its attributes. It consists of two steps: Expectation step (E-step) and Maximization step (M-step).

In EM clustering, we initialize the parameters (i.e., mean and variance) for *k* probability distributions. Then we alternate between the following E step and M step. For each of the iteration, the algorithm first executes the E-step followed by the M – step thus:

- **E-Step**—Compute the probabilities needed in the M-step, based on our current estimates of the distribution parameters.
- **M-Step**—Use the probabilities from the E step to recomputed the distribution parameters, based on maximum likelihood estimators. The algorithm terminates when the parameters converges or the algorithm reaches the maximum number of iterations [15].

REFERENCES

[1] Alosefer, Y.; Rana, O.F., (2011); "Predicting client-side attacks via behavior analysis using honeypot data," *Next Generation Web Services Practices (NWeSP), 2011 7 th International Conference on Next Generation Web Services Practices,* pp.31,36.

[2] Babu, A.R., Markandeyulu, M., Nagarjuna, B.V. (2012); Pattern clustering with similarity measures. Int. J.Comput.Technol.Appl. **3**(1), 365–369.

[3] Biau, G. (2013); "Analysis of a Random Forests Model". Journal of Machine Learning Research, 1063-1095.

[4] Chumachenko, K. (2017); Machine Learning Methods for Malware Detection and Classification

[5] Distler, D. (2007); Malware Analysis: An Introduction; SANS Institute InfoSec Reading Room

[6] Holz, T. (2009): "Tracking and Mitigation of Malicious Remote Control Network". PhD Thesis, University of Mannheim,

[7] Jain, P., and Sardana, A. (2012); "Defending against Internet Worms using Honeyfarm", Proc. CUBE International Information Technology Conference (CUBE'12), Pune, India, pp. 795-800.

[8] Kaur, R. and Singh, M. (2014); "Efficient hybrid technique for detecting zero-day polymorphic worms," Advance Computing Conference (IACC), 2014 IEEE International , pp.95,100.

[9] Kaur, R. (2016); Efficient Zero-day Attacks Detection Techniques; Computer Science and Engineering department, Thapar Univeristy, Patiala, India

[10]Kendall, K. and McMillan, C. (2007); Practical Malware Analysis. Black Hat USA – Briefing and Training.

[11]Leita, C. and Dacier, M. (2007); SGNET: A Distributed Infrastructure to Handle Zero-day Exploits, Technical Report EURECOM+2164, EURECOM institute, France

[12]Li, Z., Sanghi, M., Chen, Y., Kao, M. Y., and Chavez, B., (2006); "Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience", Proc. of the IEEE Symposium on Security and Privacy (S&P'06), Berkeley/Oakland, CA, pp. 15-47.

[13] Louppe, G. (2014). Understanding Random Forests.

[14]Mitchel, B. (2017); What is a keylogger and keylogging software? Retrieved from https://www.lifewire.com/definition-of-keylogger-817998 29th September, 2017

[15]Pai, S. (2015): A Comparison of Clustering Techniques for Malware Analysis. A publication of San Jose State University (SJSC) ScholarWorks

[16]Pai, S., Troia, F. D., Visaggio, C. A., Austin, T., Mark Stamp (2016): "Clustering for malware classification". Springer publication

[17]Prasad, B. J., Haritha, A., and Krishna S. P. (2016). "Basic static malware analysis using open source tools".

[18]Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N. (2007) "The ghost in the browser: Analysis of web-based malware". Proceedings of the 1stWorkshop on Hot Topics in Understanding Botnets (HotBots)

[19]Polychronakis, M., Anagnostakis, K. G. and Markatos, E. P. (2006); "Network-level Polymorphic Shellcode Detection using Emulation", in Journal in Computer Virology, vol. 2, no. 4, pp. 257-274.

[20]Sikorski, M., and Hoing, A. (2013); Practical Malware Analysis – The Hands-on guide to Dissecting Malicious Software. No Starch Press Publication. 1st Edition

[21] Srivastava, T. (2014); Introduction to K-nearest neigbors: Simplified. www.analyticsvidhya.com/blog/2014/10/

[22]Subrahmanian, V. S., Ovelgonne, M., Dumitras, T., Prakash, B. A., (2015); Types of Malware and Malware Distribution Strategies. "The Global Cyber-Vulnerability Report: Springer publications".

[23]Symantec (2014); "Internet Security Threat Report," Security Response Publications. vol.19. http://www.symantec.com/content/en/us/enterprise/other_resources/bistr_main_report_v19_21291018.en-us.pdf.

[24] Sophos (2014); "Security Threat Report: Smarter, Shadier, Stealthier Malware" Sophos Publications.

[25] Unuchek R (2013); The Most Sophisticated Android Trojan, June 6 2013, http://securelist.com/blog/research/35929/the-most-sophisticated-android-trojan/

[26] Wang, X., Lu, H. Zhao, B., Wang, F., and Su, J. (2013); "ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences", in Mathematical and Computer Modelling, vol. 58, no. 5, pp. 1140–1154.

[27] Weaver, N, Paxson V, Staniford S, Cunningham R (2003) A taxonomy of computer worms. Proceedings of the 2003 ACM Workshop on Rapid Malcode, WORM'03, pp 11–18, NY, USA

[28] Zeltser, L (2014); What is malware? The SANS institute publication www.securingthehuman.org/ouch

[29] https://en.wikipedia.org/wiki/Malware_analysis. (updated 10th may 2017)

[30]http://resources.infosecinstitute.com/common-social-engineering-attacks/#gref

[31]http://searchsecurity.techtarget.com/definition/metamorphic-and-polymorphic-malware (retrieved 22nd August 2017)

[32]http://cs.sru.edu/~mullins/cpsc100book/module05_SoftwareAndAdmin.html