International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:6, 2015

A Comparative Study of Virus Detection Techniques

Sulaiman Al Amro, Ali Alkhalifah

***Abstract*—**The growing number of computer viruses and the detection of zero day malware have been the concern for security researchers for a large period of time. Existing antivirus products (AVs) rely on detecting virus signatures which do not provide a full solution to the problems associated with these viruses. The use of logic formulae to model the behaviour of viruses is one of the most encouraging recent developments in virus research, which provides alternatives to classic virus detection methods. In this paper, we proposed a comparative study about different virus detection techniques. This paper provides the advantages and drawbacks of different detection techniques. Different techniques will be used in this paper to provide a discussion about what technique is more effective to detect computer viruses.

***Keywords*—**computer viruses, virus detection, signature-based, behaviour-based, heuristic-based.

I. Introduction

SINCE they first appeared, computer viruses have caused disruption to private and public organisations, governments and computer users, as they attempt to remove, modify or steal sensitive data. It is highly recommended that virus researchers should be aware of new trends, which virus writers will exploit whenever they have the opportunity. The success that attackers enjoy demonstrates that there needs to be a novel and robust detection system to prevent attacks. Therefore, a novel system is needed in order to minimise damages caused by these viruses and to defeat the new techniques used by skilful attackers. Existing antivirus (AV) products provide detection techniques which are based on signatures that have been collected from previous SEEN viruses and then added to an AV database. Prior to the arriving of a virus to the system, its signature will be compared with those stored in the database and if there is a match, the virus will be detected; otherwise, the system will run normally [1]. Thus, zero day viruses will not be detected by traditional detection systems unless this new virus is received by the antivirus company and the virus signature is stored in its own database.

Signature-based detection systems need databases in order to store the signatures. As the number of viruses increases every day, ever larger databases are needed to store all their signatures, so that more storage space will be needed in the near future [3], [2], [13]. The large database will also affect the speed of searching for signatures, and, thus, affect the performance of the system. These disadvantages mean that the signature-based detection techniques will soon be inadequate to protect computer systems.

Behaviour-based virus detection systems have been developed recently. They do not rely on a database of signatures, but instead concentrate on the behaviour of the system. They have come to light in order to overcome the problems associated with traditional signature-based detection. The principle behind this approach is first to observe the normal behaviour of the system, after which any deviation from it will be classified as an intrusion [5]. The second is to predefine virus behaviour, so that any process which resembles virus activity can be identified as a potential virus. However, there are difficulties associated with behaviour-based detection, the greatest of which is how to define the behaviours that will detect known and novel viruses without confusing them with normal processes running in the system (known as false positives). In addition, some existing virus behaviour detection techniques rely on detecting subclasses of viruses. In general, behaviour-based detection techniques rely on identifying virus characteristics in order to detect these viruses and other viruses sharing the same characteristics in the future. One of the objectives of this research is to look into the Application Programing Interface (API) calls issued by computer viruses in order to specify virus behaviour that will be used in this research.

This paper provides an overview of computer viruses. It starts with a background about computer viruses that tries to answer a number of important questions which should be asked about computer viruses, such as what they are, how they are able to spread and harm individual computers, who writes them, what they wish to accomplish and what techniques have been used to defend our systems from viruses. Then, definitions of some terms that will be significant in this research will be provided. After that, more details about computer viruses and the famous types of computer viruses will be discussed. Section V will discuss different techniques used to detect computer viruses besides their advantages and drawbacks.

II. Background

Since the late 1980s, when the first serious computer virus appeared, a war has been waged between virus generators and the antivirus community [5]. This struggle continues to this day, thanks to the daily discovery of new techniques for generating viruses and defending systems against them. In April 2006, Kaspersky [6] reported that every month, there were over 10 thousand updates to a particular antivirus (AV) program in response to the discovery of new viruses. Early in the same year, the FBI calculated the cost of computer crime for several companies and reported that computer viruses

S. A. Al Amro is with Computer Science (CS) Department, Qassim University, Buraydah, Qassim, 51452, KSA (e-mail: samro@qu.edu.sa).

A. A. Alkhalifah is with Information Technology (IT) Department, at Qassim University, Buraydah, Qassim, 51452, KSA (e-mail: a.alkhalifah@qu.edu.sa).

caused the greatest losses (67% of overall losses) [7]. Indeed, it is apparent that computer viruses have a high cost for both individuals and organisations. Therefore, dealing with them is an essential, never-ending task and research in the antivirus field is required in order to minimise the associated threats and losses.

III. Taxonomy of Malicious Software

According to [5], there are a number of terms in the field of computer viruses which might be confused, such as Trojan horse, backdoor, worm and malicious software. The following paragraphs offer definitions of these types of malware and explanations of the terminology [5], [8].

Malware is an abbreviation of the phrase 'malicious software', which can be defined as a computer program which attempts to harm the system without the knowledge of the computer user. There are several categories of malware, including worms, viruses, Trojan horses, backdoors, bombs and rootkits [1].

A Trojan horse is a program that appears to be legal and which once executed gives the attacker unauthorised remote access to a system or can be extended to download more malicious software.

A virus is a code that recursively replicates a possibly evolved a copy of itself. In other words, it is a computer program that attaches itself to other files or processes.

A worm is a program that is designed to infect host machines by individually replicating itself across networks.

Rootkits are special tools used by an attacker after breaking into a computer system, in order to obtain root-level access.

A backdoor is a program that attempts to bypass the defense system in order to gain unauthorised access to a computer.

In addition to these definitions there is another classification of malware which includes programs like adware and spyware that are not dangerous in themselves but still harm the system by reducing its performance, exposing new vulnerabilities and weakening it in ways that might affect its usability. This malware is called **grayware** [5].

Identification is not always accurate, so the performance of a computer security system should be considered in terms of the extent of false positives and false negatives. False positives occur when normal (benign) programs are identified by a defender as malicious, while false negatives are when malicious programs are not detected but rather classified as normal.

There are many subcategories of malware other than the five most common ones, which have been defined here. In order to give a clear definition and to distinguish them clearly from other malicious software, the next section offers a comprehensive examination of computer viruses.

IV. Computer Viruses

Since their emergence in the 1980s, a large number of definitions have been put forward by many researchers as to what constitutes a computer virus. Fred Cohen, who invented the technique of defence against computer viruses, defined a virus as a program that can 'infect' other programs by modifying them to include a (possibly evolved) version of itself [9]. Later, in 2005, Peter Szor claimed that the former definition is incomplete because it does not incorporate all viruses. He defines a computer virus as "A program that recursively and explicitly copies a possibly evolved version of itself" [1].

A virus must attach itself to other programs because it is unable to be executed by itself and that is one of its main characteristics [10]. Computer viruses have succeeded in satisfying the desires of their writers, especially in spreading, causing damage and bypassing detection. Nowadays, because computers have become very important to individuals, governments and organisations, computer viruses constitute a major problem of daily life and it is one of the fundamental aspects of computing that people should be aware of [1], [4].

One of the earliest papers on computer viruses was written by Cohen and Adleman [11], [12]. Cohen was the first to use the term 'virus' and using Turing machines, he also proposed the first formal definition of computer viruses [11]. Cohen states that the only way to be fully protected against viruses is by isolating the system, but notes that this cannot be practically implemented. He concludes that for a system to be secured against viruses, it must be protected from interference with both outgoing and incoming information flows.

Adleman built upon Cohen's work by inventing more formal definitions and classifications of computer viruses. His conclusion was that any program which has been infected will cause one of the three following types of damage [12]: first, impairment of the system by doing an injury to it; second, harm to the system by replicating itself in other programs; finally, producing an imitation of itself when it cannot find a file to infect.

Computer viruses have been classified as simple and complex [5]. Simple viruses have been the backbone of malicious software for the last 25 years and can be divided into three types: file viruses, boot sector viruses and macro viruses. Within these groups, a wide range of strategies are used by virus writers in order to infect files [1], [4].

Overwriting viruses: In this method, the target code will be removed by the virus and an infected file is replaced with it.

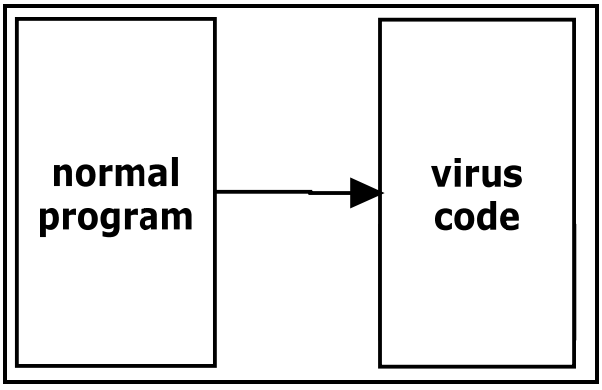

Fig. 1 Overwriting virus

Parasitic viruses: Here, a virus code will be inserted into the existing file to gain control of it. Parasitic viruses include appending and prepending types.

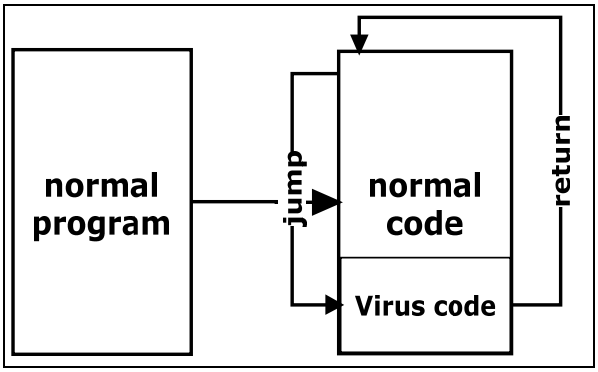

Fig. 2 Appending virus

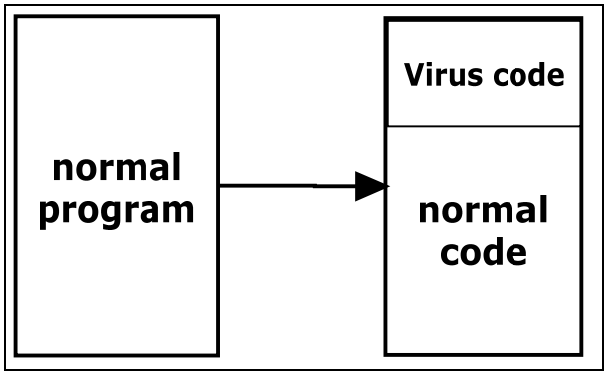

Fig. 3 Prepending virus

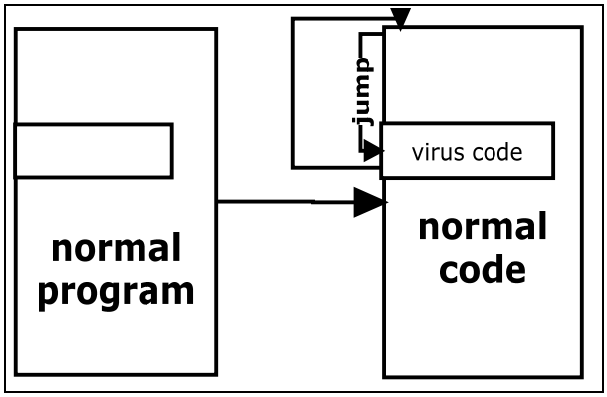

Fig. 4 Cavity virus

Companion viruses: The target file will be duplicated by a companion giving a copy of the original file that contains the virus in it.

Link viruses: A link to the virus file will be incorporated into the target file.

Application source code viruses: An active virus can be included in the source code of some applications during their installation.

Cavity viruses: These are viruses that do not increase the size of the infected file, but instead overwrite part of it by including the virus code.

Compressing viruses: As their name suggests, these viruses compress the content of the host program. The purpose of this technique is to hide the increase in the file's size after an infection has occurred.

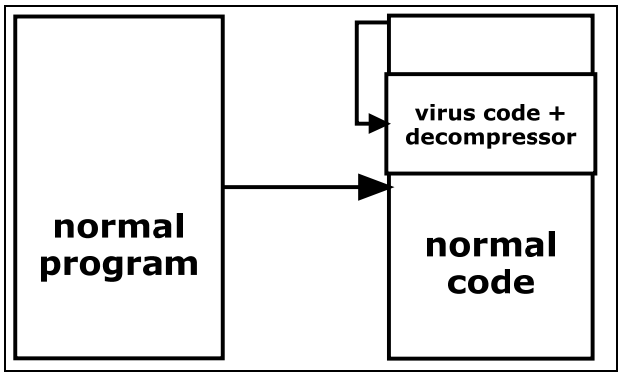

Fig. 5 Compressing virus

The master boot record, which is a type of boot sector, is normally infected with what is called a boot sector virus. The final simple virus type is the macro virus or shortcut virus, which normally repeats itself. Despite the fact that the use of these macros can be very helpful, they can also cause great damage to the system [5]. Macros can be loaded automatically when Microsoft Office applications are loaded. Therefore, the virus has an excellent opportunity to launch without notifying the user. For example, a user might receive an email contacting an attached Microsoft Word document. When the attached file is opened, the Word document launches and the macro virus is loaded on the target system. On the other hand, there are the complex or advanced viruses which have been invented by virus writers in order to evade detection techniques [1], [5]. With the evolution of defence techniques, virus writers are forced to invent viruses that are difficult for antivirus systems to detect. These can be classified into the following subcategories [1], [5]:

Encrypted viruses are encrypted in order to avoid antivirus software. This type comprises the first attempt to generate a complex virus. It was a successful technique to avoid the old signature-based detection techniques.

Oligomorphic viruses: This is the next decrypting technique which is normally detected by AV programs, where the decryption technique is randomly generated. It differs from basic encryption by having a set of decryptors rather than only one.

Polymorphic viruses: This is the most common decrypting technique ever used. The idea is that it can change its decryptors, which can take an unlimited number of forms. Polymorphic viruses have proved to be the hardest type for antivirus programs to detect.

Metamorphic viruses differ from the others in not having a decryptor; rather, they have the ability to construct new

generations that look different. The significant feature of metamorphic viruses is that they do not change the whole code, but only its functionality.

Entry-point obscuring viruses: The idea of this technique is that a code is randomly written to a location within an existing program and appears to give an update to this program. The trick is that when the trusted program is executed, the system automatically executes the virus code.

A virus can spread from program to program in the same system and can also be transferred from user to user via a network [9]. With the rapid evolution and improvement of the Internet, there have been various ways in which a virus can spread and infect systems [5], one of the best known being by email. This happens when a file is attached to a message in the mailbox; once a user clicks to open this attachment, the virus spreads. In addition to emails, downloads from the Internet, especially malicious websites, are important in spreading viruses. Removable media such as floppy disks, CDs and USBs can also cause great damage by carrying viruses in them. Users should be aware of these methods.

Computer viruses can cause low to very high damage to a system, including the removal of all information on the hard drive [1], [9], [10]. A common type of virus damage is denial of service, where the computer's resources are kept so busy that the system is unavailable to the user. Some viruses are constructed to damage certain hardware by removing all information from it (formatting), overwriting it or even destroying it. Another risk is the stealing of data from a system. Some virus writers make money by accessing individual's systems and stealing their credit card numbers and other important information in phishing attacks, using backdoor features, for example [1]. However, this research will only concentrate on those which infect other files or programs. Therefore, any virus that follows the theory of attachment and infect another file within the operating system, such malware would be targeted and detected by the present approach.

V. Computer Virus Detection

In the late 1980s and early 1990s, writing antivirus software was not very hard because at that time many individuals could create one. Two papers [11], [12] opened a path for computer virus researchers to establish a number of studies in the field of virus detection. Despite this, antivirus techniques have been developed successfully in dealing with computer viruses during the last 25 years. Virus detection techniques can be defined according to how the presence of a virus can be identified in an object [13].

A great number of detection techniques have been discussed [30], with their advantages and disadvantages. However, there are two basic detection techniques which can be distinguished, namely manual (on-demand scanning) and on-access (real-time scanning) [1], [5].

On-demand scanning is a simple virus detection technique where the user initiates the scan. This technique is not sufficient to deal with dynamic malwares such as macro viruses. In addition, it is an offline scan that cannot detect a virus unless the user is aware of it and allows scanning; otherwise, the virus will infect the system. Most AV products use this type of detection as a secondary capability [5]. The other type of scanning, which is called on-access, dynamic or real-time scanning, is a more powerful technique because of its ability to detect more complex viruses [4]. This type of detection normally happens without the knowledge of the user. The AV product scans the system memory and the hard disk looking for viruses, as the computer user browses email, opens an application or downloads cyber-content. In this technique, if a virus is detected the malicious activity will be halted, then the user will be notified and advised to take action. This type of detection is commonly used in the commercial market today.

A. Signature-Based Detection

Signature-based detection works by searching for particular sequences of bytes within an object in order to identify exceptionally a particular version of a virus [13]. Also known as string scanning, it is the simplest form of scanning, constructed upon databases which have virus signatures. When a new virus emerges, its binary form will be specifically and uniquely analysed by a virus researcher and its sequences of bytes will be added to the virus database [13]. A virus is identified by its sequences of bytes and what is called a virus signature. In addition, a hash value is another type of signatures. A large amount of data is converted into a single value by a mathematical function or a procedure known as a hash function [14].

Most AV products around the world use the signature-based technique and are trying to develop it, despite the fact that it is not sufficient for most viruses (as will be discussed later). Indeed, it has certain limitations that make it not good enough to meet the evolution and acceleration of new technologies [5]. One of its greatest weaknesses is that it is based on signature databases, which need to be updated regularly. Therefore, two actions are required: a list of signatures must be produced by the vendor, then downloaded and installed by the consumer. Another important drawback of this approach is that it needs a large database in order to store the signatures. As the number of viruses increases every day, ever larger databases are needed to store all their signatures, so that large storage space will be needed in the near future. The large database will also affect the speed of searching for signatures, and, thus, affect the performance of the system. These disadvantages mean that the signature-based detection techniques will soon be inadequate to protect computer systems [15]. In addition, many viruses today can mutate in various ways, including polymorphic and metamorphic viruses. Because signature-based detection can only identify and detect the signatures in its databases, these viruses will normally defeat the engine and bypass the defender. One of the important capabilities that signature-based detection lacks is the detection of unknown and novel viruses. For each new virus to be discovered and added to the consumer update list, antivirus software companies will take at least seven hours [16]. Meanwhile, any new virus which tries to harm the system will certainly do so without being detected.

B. Heuristic-Based Detection

The second type of on-access scanning is heuristic-based detection, which was developed to overcome the limitations of signature-based detection. While new viruses are being discovered and analysed by the AV company, before it is able to release a signature, the user has a basic defence [5]. This type of detection monitors system behaviours and keystrokes, searching for abnormal activity, rather than searching for known signatures. Thus, some AV programs that use heuristic analysis can be used and run without updating; no action is required of either the vendor or the consumer [4]. Heuristic-based detection can thus be utilised and applied without prior knowledge of computer viruses, but it has several shortcomings, one of the most annoying of which is the creation of many more false positives than signature-based systems [1]. This is less dangerous than a false negative, but nonetheless annoying to the end-user. Such systems also need more storage space and have more effect on the system performance. Their final disadvantage is that in order to perform the heuristic analysis, extra code is needed; besides a third-party component, such as protocol parsers, needs also to be included. As a result, buggier code and increase vulnerabilities [5]. While heuristic-based detection can be applied without prior knowledge of computer viruses, it still needs previous knowledge of the vulnerability [17].

Nowadays, computer virus writers have the benefit of using these packers to make their viruses run faster, as well as avoiding detection systems. Furthermore, the methods of packing make recognising and understanding viruses very complicated both for detection systems and analysts, because the authors can make small code modifications in order to change a signature and so avoid detection. Packing also makes analysis by researchers less easy, because to extract and understand unpacked code requires a third party tool, beside a deep and strong understanding of assembly language and the kernel, which leads to a better understanding of low level programming.

C. Behaviour-based Virus Detection

In behaviour-based detection, a program can be identified as a virus or not by inspecting its execution behaviour [1], [18]. Unlike traditional detection techniques which rely on signatures, in behaviour-based detection, normal and abnormal measures are used in order to determine whether or not the behaviour of a running process marks it as a virus [19]. When unusual behaviour is observed, the execution of the program will be terminated. Morales et al. [20] state that despite its drawbacks, including false positives, behaviour-based detection is still the most encouraging technique, especially in dealing with novel and anonymous viruses. Therefore, behaviour-based detection has been chosen as the topic of this research.

Ellis et al. [21] used behavioural signatures in order to improve the automatic detection of worms. Signature-based detection searches for fixed regular expressions in payloads. Instead, and at a higher level of abstraction, behavioural techniques detect patterns of executional behaviour. Ellis et al. [21] define behavioural signatures as the description of aspects of any specific behaviour of worms which are common across the manifestations of a particular worm in which its node is spanned in a temporal order. Even if a worm has not been released previously, a behavioural signature can be used to detect common implementations and the design of a worm. In general, three characteristic patterns in a network identify worm behaviour. The first is when similar data are sent between two machines, the second is when tree-like structures are observed to proliferate and the third is when a server changes into a client. Ellis et al. [21] used the notion of network application architecture (NAA), which affects the sensitivity of behavioural signatures, as an approach to distribute network applications. It is much more challenging if an attacker wants to evade the behavioural signature, because a fundamental change in behaviour is needed, rather than only in its network footprint, which is a way of knowing the system's vulnerabilities and trying to find a method to intrude into the system. In order to detect worms, [21] placed constraints on network traffic which are violated by worm traffic patterns; these violations have proven to be straightforward to detect. They used the Abstract Communication Network (ACN) model, which is a network theoretical approach to computer networks and related data flows. The NAAs, behavioural signatures and worm propagation network are all performed within the framework of the ACN. Then, in order to identify the spreading of worms across a network, the propagation of a worm is built. The result of worm spread is the capture of a communication pattern, which is identified by the offspring relation between nodes in the spanning trees of worm propagation. Two things were improved by this paper: first, the detection of worms can now be done without previous knowledge of worms; second, the work has shown an improvement in worm detection sensitivity.

Even if [21] can be considered as behaviour-based malware detection, its approach failed in the detection of unknown malware as stated by [23], [24] due to the fact that its approach relies on signatures to detect malware. Therefore, it can be said that the main purpose of behaviour-based virus detection is to detect anonymous malware which is missing in [21] approach. The other limitation is that large amounts of state information about network host behaviours need to be maintained by the behavioural techniques. This could be quite expensive in practice [25].

Later, Morales et al. [20] argue that detecting viruses in terms of their behaviour does not need any subsequent training analysis of known viruses and this means that less database will be needed; therefore, less storage space will be used. Their approach relies on detecting the behaviour of file viruses by their attempts to replicate. They apply runtime detection by monitoring executing processes that attempt to replicate. The behaviour of the virus is characterised by a property called self-replication, which happens when a process (virus) refers to itself (known as a transitive relation) during its attempt to replicate in read and write operation. Morales et al. use this property to distinguish between non-malignant processes and viruses.

Implementation is done by a runtime monitoring prototype called SRRAT, focusing on the tracking of Kernel mode system services and system user mode Win32 API calls. Despite the fact that the approach used in [20] has been shown to be good at detecting known and novel viruses without the need for prior knowledge of previous viruses, this detection technique may be bypassed by various viruses which replicate outside, across other directories within the same operating system. Furthermore, it can be argued that the definition of self-replication in Morales et al.'s approach is not complete due to the fact that their results have shown that there are a huge number of viruses in their analysis which did not follow their theory [26]. In addition, [20] approach lacks parallel detection in which they have two separate detections, one at the user level and the other at the kernel level, leaving the system too busy as claimed by [27]. The present research has the advantage of being able to observe both user level's API calls and kernel level's Native API calls at the same time.

D. Other Types of Detections

Among the many different techniques that have been used to solve the problem of computer virus detection, most have failed, have offered no advantages over existing ones or cannot be used in the real world due to their impracticability, such as file integrity checking [4]. Such unsuccessful techniques have not been discussed in this paper.

VI. Behaviour-Based Vs. Heuristic-Based detection

It has been argued that heuristic-based detection is similar to behaviour-based detection, the technique used in the present research. In fact, there is a grey area between the two detection techniques, but they differ substantially in their functionalities and ways of detecting viruses. Heuristic products check the code itself, trying to match it with known malware in order to detect new variants, whereas behaviour-based detection looks for the actions carried out by a program, intervening when it observes malevolent behaviour [18].

Behaviour-based virus detetction can be used to solve the problems associated with the heuristic-based virus detection by tracking the lists of API calls. By providing a precise definition of virus behaviours, the problem of false positives can be solved. In addition, behaviour-based detection does not require more space. As a result, less space is needed and therefore, less effect on the system performance. Furthermore, due to the fact that behaviour-based virus detection does not require a third party component, there is no need to the extra code. Therefore, the vulnerability problem associated with heuristic-based detection can be solved.

VII. Discussion

The study of computer viruses and their potential for infecting a computer system is active research, especially in the area of detecting anonymous viruses. Efficient implementation techniques have been submitted by many recent works to enhance their performance. Despite the fact that some of the submitted new ideas might enhance the detection of computer viruses based on their signature, at the same time their inability to detect novel and unknown viruses make them inappropriate for dealing with daily and new threats [15]. Even if the signature-based approaches try to deal rapidly with the unknown viruses by analysing them and updating their database, this solution is not perfect due to very expensive damage that can happen to the system during the update, and hence, the system has already been inflicted by the virus [28]. Altaher et al. [29] state that "The inability of traditional signature- based malware detection approaches to catch polymorphic and new, previously unseen malwares has shifted the focus of malware detection research to find more generalised and scalable features that can identify malicious behaviour as a process instead of a single static signature" [29]. On the other hand, heuristic-based detection techniques have not provided a good solution due to the fact that they produce many more false positives than signature-based systems. Besides, they need more storage space and have more effect on the system performance. Hence, detecting computer viruses in terms of their behaviour will help with understanding their actions, resulting in detecting unknown and newly released viruses that are a threat to computer systems every day with a better system performance.

Various frameworks have been proposed by researchers to prove that behaviour-based virus detection can deal with unknown viruses [30]-[33], but these are still hard to understand and have some disadvantages. Moreover, some of the proposed frameworks use more than one database that is updated when a new virus is received, and thus the same problem associated with traditional antivirus software aforementioned is still unsolved. However, some of these approaches concentrate on only either the user or the kernel level of Windows operating system and this single concentration might result in infecting the system. Some examples of behaviour-based virus detection will be explained in the next subsection.

A. Examples of Behaviour-Based Virus Detection

Skormin et al. [31] designed an approach that intercepts API calls while a program is running. They detect any attempt by a malware to self-replicate at run-time. Their methodology was to trace the behaviour of normal processes and analyse API calls issued by each of them along with their input, outputs argument and the execution result. The replication of a process was modelled by the Gene of Self-Replication (GSR) based upon building blocks. Each block in the GSR is considered as a portion of the self -replication process which includes seeking files and directories, writing to files, reading from files, and closing and opening a file. This approach might detect several viruses from different classes, but on the other hand, they intercepted Native API calls in the kernel. As observed by [34], [35] Native APIs are not fully documented and that means that some viruses exist which might use some of these undocumented APIs and attack the system. In addition, Skormin [26] states that "while the number of malicious computer programs that could be written is infinite, the number of ways to implement self-replication is very limited".

Later, Alazab et al. [30] used a static analysis in order to track API calls. They analysed malware to classify executable programs as normal or malicious. They plugged in the disassembler, IDA Pro [36] in their own Python program to automatically extract API calls. They examined groups of virus steps such as search, copy, delete, read and write. They found that read and write files were mostly API calls used by malwares to infect the program. However, Zwanger and Freiling [37] stated that due to the fact that Alazab et al. [30] based their approach on IDA Pro in their detection, they can only deal with user level's PE files [38]. Therefore, there are some viruses that might not be detected by [30] because they directly call the kernel by using Native API calls as mentioned by [35] and in their approach they only intercept user API calls.

Recently, Veeramani and Rai [32] used statistical analysis for Windows API calls to describe the behaviour of programs. They used an automated framework for analysing and categorising executables that rely on their relevant API calls. They tried to increase the detection rate by using a Document Class wise Frequency feature selection (DCFS) measure by getting the information related to malware from the extracted API calls. They categorised malware into groups and the relevant APIs were extracted from these categories. DCFS based feature selection measure is used to classify the executable as malicious or benign. In the [32] approach, they used a static technique to analyse malware however, as stated by Bayer [38], due to the nature of computer viruses, they can be designed to obfuscate the static analyser. Therefore, it can be said that there might be something missing during their analysis. In addition, their analysis and detection have been done at the user level leaving the system liable to viruses that can directly contact the kernel [35]. That means that [30]-[32] approaches might have a number of false positives and negatives because they rely on either kernel or user level [35], [36]. This problem can be solved by combining and tracking the Native and Win32 API calls coming from the user and kernel level that will be used in this research and explained in detail in this paper [22].

Latterly, Ravi and Manoharan [33] proposed a system which utilised Windows API call sequence. They used a statistical model called 3rd order Markov chain to model API calls. Their system comprises 3 stages: Offline, Online and Iterative learning stages. The Offline stage subsequently comprises dataset, API call tracer, API index database, signature database, rule generator and rule database. In addition, the online stage respectively comprises the target process, API call tracer, API index database and the classifier. Finally, in the iterative learning phase, after each classification, the API call sequence and the classification label of the target process is repetitively added to the signature database to enhance the training model. It can be shown that [33] used two different databases in their approach, namely, database signature and the API index database. The API calls are represented using integer IDs and then stored in the API index database. In addition, the signature database stores both the API call integer sequence and the corresponding label of all the samples in the dataset.

Ravi and Manoharan claim that their detection accuracy is better than several related approaches to their work. However, they used more than one database to store their information to catch malware. This may be acceptable as long as the detection rate is high. On the other hand, they have two main drawbacks as mentioned earlier. Firstly, they lack the detection of novel and unknown viruses which is why the behaviour-based virus detection was introduced [1], [39]. Secondly, they just intercept Windows API calls at the user level and never monitor the kernel level Native API calls, leaving their system liable to malware that directly contact the low level of Windows operating system [34]. These shortages mean that their system has no advantages over the traditional signature-based virus detection.

VIII. Conclusion and Future Work

Behaviour-based virus detection is a very topical subject area. It has been developed to overcome the problems associated with traditional signature-based virus detection. In this paper, a comprehensive description of computer viruses with the differences between them and other types of malicious software has been presented. The well-known signature-based virus detection was detailed with its pros and cons. In addition other techniques of virus detection with their positive and negative effects in computer systems have been provided. This paper has concentrated more on behaviour-based virus detection as it is the main topic of this research. Different works which have used this technique to detect computer viruses have been discussed in this paper. In addition, the system service, known as API, which can be used to analyse and trace computer viruses in this research has been discussed. Finally, related work to our research which has used this system service has been described and criticised.

We believe that by merging more than one technique, a better result can be provided. Therefore, our future work is to develop this research by examining what can be done if two or more techniques (signature, heuristic and behaviour) are used.

References

[1] Szor, P., 2005. The art of computer virus research and defense. Addison-Wesley Professional.

[2] Britt, W., Gopalaswamy, S., Hamilton, J. A., Dozier, G. V. and Chang, K. H., 2007. Computer defense using artificial intelligence, Proceedings of the 2007 spring simulation multiconference-Volume 3 2007, Society for Computer Simulation International, pp. 378-386.

[3] Harmer, P. K., Williams, P. D., Gunsch, G. H. and Lamont, G. B., 2002. An artificial immune system architecture for computer security applications. Evolutionary Computation, IEEE Transactions on, 6(3), pp. 252-280.

[4] Filiol, E., 2005. Computer viruses: from theory to applications. Springer Paris etc.

[5] Davis, M., Bodmer, S. and Lemasters, A., 2010. Hacking Exposed Malware and Rootkits. McGraw-Hill, Inc.

[6] Kaspersky, E., 2006-last update, Problems for AV vendors: Some thoughts (Homepage of Kaspersky Lab, Russia), (Online). Available: http://www.virusbtn.com/virusbulletin/archive/2006/04/vb200604-comment.dkb?mobile_on=yes (01/31, 2014).

[7] Evers, J., January 19, 2006, 2006-last update, Computer crimes cost 67 billion, FBI says (Homepage of Cnet), (Online). Available: http://news.cnet.com/2100-7349_3-6028946.html (01/31, 2014).

[8] Siddiqui, M. A., 2008. Data mining methods for malware detection. ProQuest.

[9] Cohen, F. B. and Cohen, D. F., 1994. A short course on computer viruses. John Wiley & Sons, Inc.
[10] Skoudis, E. and Zeltser, L., 2004. Malware: Fighting malicious code. Prentice Hall PTR.
[11] Cohen, F., 1987. Computer viruses: theory and experiments. Computers & Security, 6(1), pp. 22-35.
[12] Adleman, L., 1990. An abstract theory of computer viruses, Advances in Cryptology—CRYPTO'88 1990, Springer, pp. 354-374.
[13] Morales, J.A., 2008. A behavior based approach to virus detection, Florida International University.
[14] Rabah, K., 2005. Secure implementation of message digest, authentication and digital signature. Information Technology Journal, 4(3), pp. 204-221.
[15] Yoo, I. S. and Ultes-Nitsche, U., 2006. Non-signature based virus detection. Journal in Computer Virology, 2(3), pp. 163-186.
[16] Livingston, B., 23/02/2006, 2006-last update, How Long Must You Wait for an Anti-Virus Fix? - eSecurity Planet. Available: http://www.esecurityplanet.com/views/article.php/3316511/How-Long-Must-You-Wait-for-an-AntiVirus-Fix.htm (2/2/2013).
[17] Christodorescu, M., Jha, S., Maughan, D., Song, D. and Wang, C., 2006. Malware Detection. Springer.
[18] Conry-Murray, A., 2002. Behavior-blocking stops unknown malicious code. Network Magazine.
[19] Messmer, E., 01/28/02, 2002-last update, Behavior blocking repels new viruses (Homepage of Network World Fusion), (Online). Available: http://www.networkworld.com/news/2002/0128antivirus.html (02/02/2011).
[20] Morales, J. A., Clarke, P. J. and Deng, Y., 2010. Identification of file infecting viruses through detection of self-reference replication. Journal in computer virology, 6(2), pp. 161-180.
[21] Ellis, D. R., Aiken, J. G., Attwood, K. S. and Tenaglia, S. D., 2004. A behavioral approach to worm detection, Proceedings of the 2004 ACM workshop on Rapid malcode 2004, ACM, pp. 43-53.
[22] S. Al Amro, A. Cau, "Behaviour-based virus detection system using Interval Temporal Logic," Proceedings of the 6th IEEE International Conference on Risks and Security of Internet and Systems, pp.1-6, Sept. 2011.
[23] Chiang, H. and Tsaur, W., 2010. Mobile Malware Behavioral Analysis and Preventive Strategy Using Ontology, Social Computing (SocialCom), 2010 IEEE Second International Conference on 2010, IEEE, pp. 1080-1085.
[24] Idika, N. and Mathur, A.P., 2007. A survey of malware detection techniques. Purdue University, pp. 48.
[25] Zhang, Q., 2008. Polymorphic and metamorphic malware detection. ProQuest..
[26] Skormin, V.A., 2010. Server Level Analysis of Network Operation Utilizing System Call Data. Binghamton Univ New York Dept of Electrical and Computer Engineering. Blade API Monitor. http://www.bladeapimonitor.com/, 2011.
[27] BOS, H., 2013-last update, D16 (D4. 2) Analysis Report of Behavioral Features (Homepage of Wombat), (Online). Available: http://www.wombat-project.eu/WP4/FP7-ICT-216026-Wombat_WP4_D16_V01_Analysis-Report-of-Behavioral-features.pdf (12/20/2012).
[28] Moskovitch, R., elovici, Y. and Rokach, L., 2008. Detection of unknown computer worms based on behavioral classification of the host. Computational Statistics & Data Analysis, 52(9), pp. 4544-4566.
[29] Altaher, A., Ramadass, S. and Ali, A., 2011. Computer virus detection using features ranking and machine learning. Australian Journal of Basic and Applied Sciences, 5(9), pp. 1482-1486.
[30] Alazab, M., Venkataraman, S. and Watters, P., 2010. Towards Understanding Malware Behaviour by the Extraction of API Calls, Second Cybercrime and Trustworthy Computing Workshop 2010, pp. 52-59.
[31] Skormin, V., Volynkin, A., Summerville, D. and Moronski, J., 2007. Prevention of information attacks by run-time detection of self-replication in computer codes. Journal of Computer Security, 15(2), pp. 273-302.
[32] Veeramani, R. and Rai, N., 2012. Windows API based Malware Detection and Framework Analysis. International Journal of Scientific & Engineering Research (IJSER), 3(3).
[33] Ravi, C. and Manoharan, R., 2012. Malware Detection using Windows API Sequence and Machine Learning. International Journal of Computer Applications, 43(17), pp. 12-16.
[34] Seifert, C., Steenson, R., Welch, I., Komisarczuk, P. and Endicott-Popovsky, B., 2007. Capture–A behavioral analysis tool for applications and documents. Digital investigation, 4, pp. 23-30.
[35] Russinovich, M., 2011-last update, Inside the Native API (Homepage of Sysinternals), (Online). Available: http://www.sysinternals.com/Information/NativeApi.html (1/22/2014).
[36] Rescue, D., 2006. IDA Pro Disassembler. 2006-10-20. http://www.datarescue.com/idabase.
[37] Zwanger, V. and Freiling, F.C., 2013. Kernel mode API spectroscopy for incident response and digital forensics, Proceedings of the 2nd ACM SIGPLAN Program, Protection and Reverse Engineering Workshop 2013, ACM, pp. 3.
[38] Bayer, U., Moser, A., Kruegel, C. and Kirda, E., 2006. Dynamic analysis of malicious code. Journal in Computer Virology, 2(1), pp. 67-77.
[39] Jacob, G., Debar, H. and Filiol, E., 2008. Behavioral detection of malware: from a survey towards an established taxonomy. Journal in Computer Virology, 4(3), pp. 251-266.