Extended essay cover

Candidates must complete this page and then give this cover and their final version of the extended essay to their supervisor.

Candidate session number

Candidate name

School name

Examination session (May or November) | MAY | Year | 2015

Diploma Programme subject in which this extended essay is registered: **Computer Science**
(For an extended essay in the area of languages, state the language and whether it is group 1 or group 2.)

Title of the extended essay: **How Effective are Antivirus Programs at Detecting Ever-Evolving, 'Complex' Virus Threats?**

Candidate's declaration

*This declaration must be signed by the candidate; otherwise a mark of zero will be issued.*

The extended essay I am submitting is my own work (apart from guidance allowed by the International Baccalaureate).

I have acknowledged each use of the words, graphics or ideas of another person, whether written, oral or visual.

I am aware that the word limit for all extended essays is 4000 words and that examiners are not required to read beyond this limit.

This is the final version of my extended essay.

Candidate's signature: ___________________________ Date: 15-01-2015
Supervisor's report and declaration

The supervisor must complete this report, sign the declaration and then give the final version of the extended essay, with this cover attached, to the Diploma Programme coordinator.

Name of supervisor (CAPITAL letters) ___________________________

Please comment, as appropriate, on the candidate's performance, the context in which the candidate undertook the research for the extended essay, any difficulties encountered and how these were overcome (see page 13 of the extended essay guide). The concluding interview (viva voce) may provide useful information. These comments can help the examiner award a level for criterion K (holistic judgment). Do not comment on any adverse personal circumstances that may have affected the candidate. If the amount of time spent with the candidate was zero, you must explain this, in particular how it was then possible to authenticate the essay as the candidate's own work. You may attach an additional sheet if there is insufficient space here.

Student name: ___________________________ Supervisor name: ___________________________

B/A grade EE: ___________________________

This declaration must be signed by the supervisor; otherwise a mark of zero will be issued.

I have read the final version of the extended essay that will be submitted to the examiner.

To the best of my knowledge, the extended essay is the authentic work of the candidate.

As per the section entitled "Responsibilities of the Supervisor" in the EE guide, the recommended number of hours spent with candidates is between 3 and 5 hours. Schools will be contacted when the number of hours is left blank, or where 0 hours are stated and there lacks an explanation. Schools will also be contacted in the event that number of hours spent is significantly excessive compared to the recommendation.

I spent [4] hours with the candidate discussing the progress of the extended essay.

Date: 28/2/2015
How effective are antivirus programs at detecting ever-evolving, 'complex' virus threats?
Abstract

This paper provides a very basic answer to the question "How effective are antivirus programs at detecting ever-evolving, 'complex' virus threats?" by examining the nature and definition of a virus, and also methods (like code obfuscation) that metamorphic viruses utilize to mutate and conceal themselves from antivirus programs. General detection techniques, like signature based detection are also analyzed in the paper – This includes discussions of their effectiveness at detecting metamorphic viruses but also any general strengths and weaknesses that each technique boasts. Similarly, methods such as the Hidden Markov Model that are specific to detecting metamorphic viruses will also be discussed. Additionally, trends for future metamorphic viruses are explored, indicating the possible exponential growth in destructive power of future metamorphic viruses. This paper was written mainly with the aid of the World Wide Web, with the exception of help from a few papers written specifically to discuss metamorphism and metamorphic viruses. Of course, primary research could be useful in determining the effectiveness of an antivirus program, but the difficulty of writing a metamorphic virus or even obtaining the resources to write one is beyond the capabilities of this paper. With all factors considered, this paper concludes that current antivirus programs perform fairly poorly against metamorphic viruses without the implementation of metamorphic specific detection methods. This is worrying, considering metamorphic viruses will only become increasingly common in the near future, due to the rapid evolution of virus code.
Contents

1 Introduction

1.1 Introduction ................................................................. 5

2 Introduction to Viruses

2.1 What are computer viruses? ........................................ 6
2.2 'Complex' Viruses........................................................ 6

3 Evolution of Virus Code

3.1 Encrypted Virus............................................................ 8
3.1 Oligomorphic Virus....................................................... 8
3.3 Polymorphic Virus....................................................... 9

4 Metamorphic Virus

4.1 Metamorphic Virus....................................................... 10
4.2 Code Obfuscation......................................................... 10
  4.2.1 Garbage Code......................................................... 11
  4.2.2 Instruction Replacement........................................ 11
  4.2.3 Jump Instructions.................................................. 11
  4.2.4 Permutations........................................................ 12

5 Antivirus Programs

5.1 What are antivirus programs? ...................................... 13
5.2 Generic methods of detection..................................... 13
  5.2.1 Signature based detection...................................... 13
5.2.2 Anomaly based detection ......................................................... 15
5.2.3 Emulation based detection ......................................................... 17
5.2.4 Are these methods effective at dealing with specifically, metamorphic viruses? ............................................................ 18
5.2.5 Why introduce these intrusion detection systems if they cannot detect metamorphic viruses? ....................................................... 19
5.3 Metamorphic specific detection techniques ................................. 19
5.3.1 Heuristics Emulator Based Detection ....................................... 19
5.3.2 Hidden Markov Model (HMM) .................................................. 20

6 The Future of Metamorphic Viruses

6.1 Predicting the future of metamorphic viruses and their capabilities ..................... 22

7 Conclusion

7.1 Conclusion ..................................................................................... 24
7.2 Bibliography .................................................................................. 26
7.3 Appendices ................................................................................... 29
7.3.1 Appendix A: XOR ..................................................................... 29
7.3.2 Appendix B: Jump Instructions ..................................................... 30
7.3.3 Appendix C: Permutations ............................................................ 31
7.3.4 Appendix D: Table of AVG and Avast against polymorphic virus families ............................................................................. 32
Chapter 1

Introduction

1.1 Introduction

In 2012, the number of malware infected PC's in China reached a sky high of 54.89% [11]. With such swift development into destructive viruses, research into virus detection techniques has become more important and prevalent in order to prevent malware interfering from our computers operation. "How effective are antivirus programs at detecting ever-evolving, complex virus threats?" thus becomes a question of certain significance, as it can show us the ways in which antivirus programs detect and prevent viruses from entering our computers, but more importantly, highlight if further development on antivirus programs is necessary or even worthwhile.

Personally, the question is of interest to me as I have been a victim of computer viruses which eventually piqued my interest to learn more about the ways of preventing them. This essay will attempt to analyse the effectiveness of antivirus programs by looking at the ways in which complex viruses (specifically metamorphic viruses) function and mutate (through code obfuscation) in order to evade detection by antivirus programs, and the methods antivirus programs employ to detect virus code.
Chapter 2
Introduction to viruses

2.1 What are computer viruses?

Computer viruses are malicious software (malware) designed to endlessly replicate and place copies of itself into a user’s (host’s) computer. Viruses usually operate in order to bring detrimental effects on the user, such as to steal the user’s information or to eat up their hard disk space. However, there is no defining set of rules that viruses must pertain to in terms of their destructive behaviour; the only defining feature of all computer viruses are that they undergo replication whilst placing themselves in a user’s computer without their consent.

2.2 ‘Complex’ viruses (Reference: [13])

Flame was a complex attack toolkit discovered in 2012 which aimed to steal valuable user information including personal data, image files, and even Skype conversations. Kaspersky claims it is one of the most complex and sophisticated threats ever discovered and exceeds any other cyber weapons ever created. It is interesting to note, however, that Flame had a size of 20 megabytes, but yet still remained undiscovered until 2012. Reports from Kaspersky exhibit that Flame was last ‘out in the wild’ in February 2010.

A ‘complex’ virus is essentially a virus with strong capability to mutate and remain hidden. Flame is a prime example of this – some even believe that Flame is a metamorphic virus due to its large size.
This paper will attempt to explore the effectiveness of antivirus programs against a type of virus which is considered complex and hard to cope with – Metamorphic viruses.
Chapter 3
Evolution of Virus Code

3.1 Encrypted Virus [8] [15]
One of the first methods that virus programmers utilized to hide their virus was known as encryption. Virus programmers exploited the fact that encryption allowed the virus code to be distorted into a different form, masking any notable features in the code's appearance, rendering methods that observe the presentation of the code (signature based detection) ineffective. A decryption module is also created, allowing the virus code regain its original functionality. This was usually done through a method utilizing the XOR command, because XOR-ing twice will result in the code in its original form (see Appendix A for a comprehensive explanation). However, the virus could still be identified provided the decryption key was long and unique enough.

3.2 Oligomorphic Virus [8] [15]
Oligomorphic Viruses were created in order to combat the weakness of encrypted viruses – since encrypted viruses were still easily detected by scanners due to their unique decryptor, virus developers created oligomorphic viruses with multiple decryptors, with one randomly selected for each use. The first known oligomorphic virus was known as the Whale virus [8], which included a few dozen set of fixed decryption keys.
3.3 Polymorphic Virus [8] [15]

Since oligomorphic viruses were still detectable with the small number of decryption keys, virus programmers introduced polymorphic viruses. With polymorphic viruses, the virus code is still encrypted and decrypted in turn, but the decryption key is changed on each infection. This is done with the help of a mutation engine, a polymorphic engine, which can generate several millions or even billions of differentiated decryption routines. This is unlike oligomorphic code, where only a preset amount of fixed decryption routines can be used.
Chapter 4

Metamorphic Virus

4.1 Metamorphic Virus [5] [8] [15] [16]

Metamorphic viruses are viruses that came after the era of polymorphic viruses - they have the ability to completely rewrite their program on each execution. It is important to distinguish between metamorphic viruses and polymorphic viruses – In polymorphic viruses, the same code takes on different forms through encryption; in metamorphic viruses, the code actually completely mutates, so the code is different each time it is executed. Similarly, metamorphic viruses require a metamorphic engine to transform into their metamorphic form.

Metamorphic viruses are considered one of the hardest viruses to deal with due to their complexity to program and even understand. Since metamorphic code has limited commercial use, most programmers considered it more practical to invest time and effort elsewhere [5]. Programmers who have the sufficient skill to program metamorphic code are rare and will usually be employed by security companies and antivirus companies – it is considered a very elitist skill, and thus only the more high-end anti-virus programs can detect this type of virus (In fact, this is still debatable).

4.2 Code Obfuscation [8] [17]

Metamorphic viruses easily escape detection due to the wide variety of techniques they perform to in order to conceal themselves. These techniques are known as code
obfuscation. Some of the simpler ways code obfuscation is performed are listed below:

4.2.1 Garbage Code [8]

Metamorphic viruses can be evolved by inserting 'junk code' into sections of the code. Implementing 'junk code' will act as a way of diverting the antivirus, disallowing a search on useful hexadecimal string. The junk code is essentially useless and will not interfere with the operation of the code itself.

4.2.2 Instruction Replacement [8]

Equivalent instructions can also act as a replacement to represent the same course of action that will be taken by the code. This was evident in the Zperm virus - it was able to replace the instruction ‘xor eax, eax’ with “sub eax,eax” [8]. Both of these instructions have the same functionality of zeroing the register ‘eax’, but have different code representations (or operation code, “opcode”).

4.2.3 Jump Instructions [8]

In addition to instruction replacement, the programmer can also implement a set of jump instructions. Jump instructions will allow the modification of the program counter, and allow execution of the code on the new counter (see appendix B). This allows different permutations of the code - Each time the code is executed, it is executed in a different order. This was again prevalent in the Zperm virus [8].
4.2.4 Permutations [6] [8]

If permutation can be implemented into virus code, the number of possible code generations vastly increases, significantly expanding the chances of avoiding detection. Complex permutations can be achieved through a combination of the above techniques, such as merging jump instructions and garbage code. The Ghost virus exploited this to a very high extent – Since it had the ability to reorder any of its sub-routines, and had 10 sub-routines, this could have led to the possible generation of 3628800 (or 10!) different virus generations [8]. (A more comprehensive discussion of permutations is also included in Appendix C)
Chapter 5

Antivirus programs

5.1 What are antivirus programs?

As virus programmers became more adept at programming viruses with greater destructive power, people began to find ways to combat these viruses. Antivirus programs are software developed to detect, prevent and remove computer viruses.

5.2 Generic methods of detection

There are numerous ways in which antivirus programs detect viruses. This paper will explore the methods: signature based detection, anomaly based detection, and emulation based detection.

5.2.1 Signature based detection [2] [7] [9] [19]

One common way of detecting viruses is known as the signature based detection method. Each virus has an algorithm or hash known as a signature by which it can be identified. When scanning for a virus, the antivirus will compare contents of the scanned files with a list of known signatures from its library. If a signature of a virus is detected, it will perform actions to protect the computer from further harm, by, for example, quarantining and encrypting the file, rendering it useless. The contents of the library are constantly updated with new found signatures to combat new viruses. Since the signature can be equivalent in a number of different viruses, this makes signature based detection a generic detection method, allowing it to detect new viruses of the same "family" (viruses which share the same characteristics).
An upside of signature based detection is that it is highly effective and reliable approach for detecting viruses of the same family due to the certain nature of a signature, which is always equivalent and does not employ guesswork, unlike other intrusion detection systems.

The effectiveness of signature based detection is, however, debatable against completely new viruses, or a zero-day virus, because they have no known signature. This is exemplified using set theory [7]:

Let:

\[ S = \text{Number of viruses with known signatures} \]
\[ V = \text{Number of unknown viruses} \]
If $V$ denotes the number of undiscovered viruses, and $S$ the number of viruses with known signatures, the insignificance of the number of known viruses compared to the number of unknown viruses is highlighted – It is claimed that antivirus scanners have only managed to uncover 2% of all viruses [9], and thus a method that requires an application of the features of previous viruses will be severely inept at detecting new viruses.

Another issue with this approach is of the large number of signatures that are produced daily. Mikko H. Hypoonen, chief research officer of Finnish based antivirus vendor F-Secure summarizes this: "Signatures have been dying for quite a while. The sheer number of malware samples we see every day completely overwhelms our ability to keep up with them." This is further exemplified when Symantec claims they only produced approximately 15,000 antivirus signatures daily in 2009, which has currently risen to 25,000 a day.

### 5.2.2 Anomaly based detection [7] [12]

Anomaly based detection is a two-step method used to detect viruses. A system is established where a set of rules denoting ‘normality’ is learned by the system (through artificial intelligence techniques or mathematical models). This system is then used to monitor any activity performed by the computer, flagging any suspicious activity as an anomaly. For example, the computer will formulate a sense of ‘normality’ by observing multiple URLs for their URL character length and flag any URLs that diverge off the average URL length.
Anomaly based detection is a better candidate (over signature based detection) for preventing zero-day viruses due to the wide range of anomalies that can be flagged. As such, the anomaly based detection is a potential way of preventing a wide array of viruses.

An obvious disadvantage with this method is the difficulty of training a good, reliable system. If the data used to determine ‘normality’ is infected or impure, the impurity of the data will be treated as ‘normal’ by the system in future occurrences. This can allow viruses with the similar impurities to slip by the system undetected. The opposite can also occur – if data that is too ‘strict’ is used, it is likely that the system will flag multiple clean websites as an anomaly, leading to a high false positive rate. The reliability of the system will depend greatly on the quality of the data, and as such, the collection of pure data is of utmost importance.

Additionally, unlike signature-based detection, anomaly-based detection may miss several known viruses, because it is based on observation rather than something easily identifiable (like signatures). Moreover, it can also miss any attacks that don’t extend too far off the average observation.
5.2.3 Emulation based detection [14]

Emulation based detection refers to a method in which the instructions of the virus are emulated, and then executed in a safe, virtual environment known as a sandbox. It is then examined periodically, or alerted whenever a specific instruction is executed. It is incredibly powerful, capable of detecting polymorphic and less metamorphic viruses. A downfall of this method would be the cost of implementation — the amount of resources needed to execute all lines of code of a virus is much higher than a simple scan for a signature. Another problem facing current antivirus programs is their compatibility with a type of interactive scanning detection method — some antivirus programs are simply not built with the mindset of being compatible with an emulator.

A table summarizing the key strengths and weaknesses of each detection method:

<table>
<thead>
<tr>
<th>Detection Method</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature Based</td>
<td>-Highly effective against viruses of the same family</td>
<td>-Ineffective against zero-day threats</td>
</tr>
<tr>
<td>Anomaly Based</td>
<td>-Viable method of detecting zero-day threats</td>
<td>-Takes time/money to set up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-‘Impure’ learning by system is exploitable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-High rate of false positives</td>
</tr>
<tr>
<td>Emulation Based</td>
<td>-High potential/powerful</td>
<td>-Costly to set up</td>
</tr>
</tbody>
</table>
5.2.4 Are these methods effective at detecting specifically, metamorphic viruses? [4]

Signature based detection is a method obviously incapable of detecting any type of metamorphic virus; Polymorphic viruses were already able to hide their signature through encryption and even more through mutating decryption keys, whilst metamorphic viruses alter their code to make signature detection impossible. Despite its capability at detecting viruses of the same family, with zero-day threats becoming increasingly potent by including metamorphic features, signature based detection is beginning to be questioned even by antivirus vendors due to lacking success at detecting complex viruses.

Anomaly based detection is perhaps a step up of signature based detection. Arguably, it can detect new viruses or even metamorphic viruses, but yet, its reliability and accuracy is still challenged. The antivirus vendor must factor in the ‘impure’ learning that can occur throughout the whole learning process, or else multitudes of viruses will slip by the system due to its ‘impure’ state. In fact, virus programmers can adapt the system to their needs by slowly feeding the system ‘impure’ material, changing its understanding of ‘normality’. Metamorphic viruses will also be able to bypass anomaly detection through code obfuscation methods such as garbage code insertion and jump instructions, making the system believe the virus is ‘normal’.

Emulation based detection is the most effective technique listed, since it is capable of detecting both polymorphic viruses and metamorphic viruses. In fact, Symantec endorses this method as a necessity for all future antivirus programs: “It is evident that all antivirus scanners need to go in the direction of interactive scanning engine
developments." [14]. Emulation is very effective for detecting polymorphic viruses, since encryption will not protect the tracing of their code; however, sophisticated metamorphic viruses are combatted by code obfuscation techniques that prevent emulation based detection.

5.2.5 Why introduce these intrusion detection systems if they cannot detect metamorphic viruses?

The above demonstrates how simple and even polymorphic viruses are detected by generic intrusion detection systems. Metamorphic viruses are of a higher caliber than other viruses, and therefore require specific detection methods. Most of these detection methods, however, stem off the generic intrusion detection systems listed above. Many of these generic intrusion systems are still in use however, demonstrating how antivirus programs are still utilizing old, ineffective detection methods whilst virus code is only improving and evolving.

5.3 Metamorphic specific detection techniques

Due to the incapability of intrusion detection systems a, several other methods have been developed to detect them. Listed are two of these techniques: Heuristics Emulation based detection and the Hidden Markov Model.

5.3.1 Heuristics Emulation based detection [8]

Heuristics Emulation based detection is a branch of emulation based detection, but rather, it uses heuristics, or a specialized set of rules, to define its method of detection. For example, a set of rules may include:
Session Number:

- Scanning for anything that decrypts itself during execution
- Scanning for any previously known viruses
- Scanning for anything that replicates itself into other programs

Similar to emulation based detection, any possible threats are then run in a sandbox to identify virus code or malware. Emulation is effective at detecting metamorphic viruses due to its ability to disassemble techniques such as garbage code, but viruses are able to employ anti-emulation techniques to combat detection.

Anti-emulation techniques are especially effective against heuristics emulator based detection — since it is defined only by a set of principles, viruses can be adapted to avoid these principles. For example, if an antivirus program was programmed to periodically scan for a virus every 3 hours starting from 00:00, and a virus was coded so it would replicate every day at 02:00, the antivirus program would miss the virus. The Magistr virus exploited this to a high extent — it would only replicate under the circumstances of an active internet connection, rendering it almost undetectable by emulation means when there was no internet connection. It is thus evident that we cannot rely on heuristics emulation as a way to uncover viruses, unless antivirus vendors discover a way of bypassing anti-emulation techniques.

5.3.2 Hidden Markov Model (HMM) [18]

The Hidden Markov Model refers to a method of statistical analysis used in computational biology, which has also recently been put forward as a method of virus detection. The HMM method is based on the assumption that the Markov property (that is, that the probability of an event is based solely on its present state and not on
any of its past states) exists in a continuous probability function. For a Markov process to be a Hidden Markov Model, however, at least some of the states have to be hidden or unobserved.

Essentially, data can be inserted into HMM so it can be trained where each state exemplifies properties of a virus family. Input data known as observation sequences can now be compared to the training data. The comparison is given a score, known as log likelihood per opcode (LLPO), and the higher the score, the more similar the sequence is to the training data.

The HMM method is useful at detecting viruses of the same family, since it looks for similarities in the virus code. Metamorphic viruses do not escape the HMM method either — although metamorphic viruses take on different forms in each generation, there will still be similarities between the original code and its new generations.

It boasts a very successful detection rate, too. 37 metamorphic viruses were scanned by 3 different anti-virus programs (avast! Antivirus version 4.7, eTrust version 7.0.405, AVG Anti-Virus version 7.1) and then analyzed through the HMM method. Avast! and eTrust both detected 17 viruses, while AVG detected 27, but none detected the 10 viruses that were created by Next Generation Virus Creation Kit (NGVCK), which was considered the best virus generator that was tested [18]. The HMM model, however, was able to detect all 10 of the NGVCK metamorphic viruses.
Chapter 6

The future of metamorphic viruses

6.1 Predicting the future of metamorphic viruses and their capabilities [7]

Antivirus programs are not the only piece of technology that is continually evolving; viruses writers are always looking for ways to expand the potential of their viruses.

Because metamorphic viruses are so difficult to write, and the lack of powerful metamorphic engines in existence may perhaps imply metamorphic viruses don’t currently pose a large threat to society and antivirus. In fact, many virus writers are returning to polymorphic viruses and trying to implement more code obfuscation techniques to create stronger polymorphic viruses. It is only a matter of time, however until people return to creating increasingly powerful metamorphic viruses.

One very obvious, general way of improving a virus’s capability is through collaboration. The Hybris worm/virus that appeared in 2000-2001 was written by a group of Brazilian, Spanish, French and Russian accomplished virus coders [14]. If a group of highly proficient virus developers contributed to writing a single metamorphic virus, the result would be a self-propagating virus uninterpretable to experienced antivirus researchers.

From a technical more viewpoint, perhaps metamorphic viruses can incorporate the idea of encryption and polymorphism. If a virus can be created with both polymorphic
and metamorphic features, the virus will be incredibly hard to detect due to the constant mutating code plus the possible millions of different decryption keys.

New combinations of code obfuscation methods can also be invented. For example, permutations can be incorporated with anti-emulation techniques in order to further prevent emulation. If anti-virus programs started to adapt to the Magistr virus by detecting it under an active internet connection, a new set of rules could force the virus to only replicate on Sunday's, rendering it even more avoidable to antivirus scanners.

Metamorphic engines will also be upgraded, which will not only lead to the existence of stronger metamorphic viruses, but the existence of more metamorphic engines will make metamorphic viruses more accessible to virus programmers new to metamorphism.
7.1 Conclusion

Logically, viruses most probably will have the upper hand — Antivirus programs will always be the one to react, implying that viruses will have an endless amount of time to evolve and mutate into more destructive and undetectable forms, whilst antivirus programs will only have a limited time period to discover new ways of detection and protection.

This leads us to consider, firstly, if any generic intrusion detection methods are useful in detecting complex metamorphic viruses. The above has proven they are not — Signature based detection is ineffective, anomaly based detection is viable but exploitable, and emulation-based detection is combatted by code obfuscation techniques.

We also have to consider that metamorphic specific detection methods may still be in development, or simply be avoidable. The heuristics emulation based detection method can although be effective, is already bypassed using anti-emulation techniques. The HMM method is highly effective, but is still being tested and therefore not put in commercial use.

The effectiveness of antivirus program against metamorphic viruses also varies from each vendor, seen in the data presented during the comparison of the HMM method with commercial antivirus programs: avast! and eTrust achieved a mere 46% (17/37)
detection rate, while the AVG achieved a better 72% (27/37) detection rate. This, in comparison to detection results of polymorphic viruses [1] (see Appendix D), was poor (avast! and AVG both achieved over 90% against most polymorphic virus families). As such, the poor detection rates of avast! and eTrust (against metamorphic viruses specifically) lead us to believe the ineffectiveness of these antivirus programs can only be resolved through specific detection methods. The detection rate of AVG is arguably better, but still not optimal.

Yet, the ideas of virus evolution that are presented in the ‘predicting the future of metamorphic viruses and their capabilities’ show that metamorphic viruses will evolve as well, even as new detection techniques are implemented. As such, antivirus programs will always only be playing the ‘catch up game’.

Unfortunately, obtaining information about the detection techniques the individual antivirus programs employ proves to be a difficult task, due to the risk of production of malware that could potentially target a specific antivirus program because of their known routines and methods. It is thus difficult to accurately determine the effectiveness of specific antivirus programs against metamorphic viruses.

In conclusion, from the ineffective detection methods, poor to mediocre detection rates, rapid evolution of viruses, and lack of implementation of metamorphic specific detection methods, I believe that antivirus programs are currently ineffective at detecting ever-evolving, ‘complex’ viruses.
7.2 Bibliography


Session Number:

San Jose State University.


7.3 Appendices

7.3.1 Appendix A: XOR [10]

XOR, or exclusive or, refers to a logical operation where only one input out of two yields true. This can be depicted in a simple truth table using binary numbers:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A XOR B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

As seen, the XOR gate is only true when both inputs differ and when one is true, whilst the other false. The XOR gate is often used as a method of encryption due to the reversible nature of XOR. If we consider the second truth table below:

<table>
<thead>
<tr>
<th>A XOR B</th>
<th>B</th>
<th>A XOR B XOR B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

As seen above, \( A \ XOR \ B \ XOR \ B \) is equivalent to \( A \), implying that performing the XOR function twice will reverse the encryption process, allowing viruses to return to
their original form. However, this only occurs if decryptor key (or B in this case) is known, which made it (in theory) hard for antivirus programs to detect. This was unfortunately countered by the long decryptor keys which were unique enough to be detectable.

7.3.2 Appendix B: Jump Instructions [3]

The machine instruction cycle depicts the basic functions of the central processing unit (CPU). There are 4 core steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The next instruction is retrieved from the program counter (PC), which stores the address of the instruction to be executed next.</td>
</tr>
<tr>
<td>2</td>
<td>Decode the instructions so they are readable and executable by the computer</td>
</tr>
<tr>
<td>3</td>
<td>Execute the decoded instructions</td>
</tr>
<tr>
<td>4</td>
<td>Store results into memory</td>
</tr>
</tbody>
</table>

The cycle then repeats itself by returning to step 1 again.

When a jump instruction is executed (right before step 3), it loads a new address into the PC. The cycle will then return to step 1 with the new address and repeat itself.
7.3.3 Appendix C: Permutations [6]

Permutations are a field of mathematics known as *combinatorics*, which revolves around the choosing a number of objects out of a certain arrangement of objects. In 4.2.4 *Permutations*, the Ghost virus had 10 routines that could be reordered, which, logically, can be exemplified like this:

<table>
<thead>
<tr>
<th>Routine Selected</th>
<th>Routines Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

All 10 routines are obviously available for the selection of the 1st routine, leaving only 9 routines after the first routine is selected. After the second selection, only 8 will be left. This will continue until there are no more routines available (since there is no replacement). As such, the total number of arrangements of routines is $10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 10!$, implying that the number of permutations can be exemplified by the general term $n!$. 
Another type of permutation that could be even more powerful than the above develops when replacement does occurs, much like in combination locks.

Combination locks usually have 10 numbers (0-9) on each digit – if we take a 3 digit combination lock as an example:

<table>
<thead>
<tr>
<th>Digit</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

This shows that the number of arrangement of numbers is \(10 \times 10 \times 10 = 10^3\), which can be generalized to \(n^r\). However, the problem of implementing this type of permutation lies in the fact that replacement is allowed and code functionality is mostly unexchangeable, and hence several sections of the virus code could be implemented with the same functionality. A probable way of implementing this type of permutation could utilize trash instructions, since they have no functionality at all.

### 7.3.4 Appendix D: Table of AVG and Avast against polymorphic virus families [1]

<table>
<thead>
<tr>
<th>Antivirus product \ Virus family</th>
<th>Allaple.1</th>
<th>Allaple.2</th>
<th>Allaple.3</th>
<th>Allaple.4</th>
<th>Alman.1</th>
<th>Alman.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avast</td>
<td>99.96%</td>
<td>99.00%</td>
<td>99.32%</td>
<td>93.01%</td>
<td>99.90%</td>
<td>100%</td>
</tr>
<tr>
<td>AVG</td>
<td>100%</td>
<td>99.90%</td>
<td>100%</td>
<td>99.75%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>