On Analysis and Effectiveness of Signature Based in Detecting Metamorphic Virus

Imran Edzereiq Kamarudin[1], Syahrizal Azmir Md Sharif[1] and Tutut Herawan[2]

[1]*Faculty of Computer System and Software Engineering*
Universiti Malaysia Pahang
Lebuh Raya Tun Razak, Gambang 26300, Kuantan, Pahang, Malaysia
[2]*Department of Mathematics Education*
Universitas Ahmad Dahlan
Jalan Prof Dr Soepomo 55166, Yogyakarta, Indonesia

{edzereiq,syazmir}@ump.edu.my, tutut81@uad.ac.id

Abstract

Computer viruses and other forms of malware have viewed as a threat to any software system. They have the capability to deliver a malicious infection. A common technique that virus writers use to avoid detection is to enable the virus to change itself by having some kind of self-modifying code. This kind of virus is commonly known as a metamorphic virus, and can be particularly difficult to detect. Metamorphic viruses have a potential to avoid any signature-based detection schemes by implementing code obfuscation techniques in an effort to defeat it. In metamorphic virus, if dead code is added and the control flow is changed sufficiently by inserting jump statements, the virus cannot be detected. In this paper we first developed a code obfuscation engine. We then used this engine to create metamorphic variants of a seed virus and performed the validity of the statement about metamorphic viruses and signature based detectors. Last but not least, we have propose a profile which enclose the information about the existing metamorphic viruses infection.

Keywords: *Viruses; Metamorphic viruses; Signature based detectors*

1. Introduction

Computer viruses can trace their pedigree to John von Neumann's studies of self-replicating mathematical automata in the 1940s. Although the idea of programs that could infect computers appears in 1970s, the first well documented case of a computer virus spreading "in the wild" occurred in 1986 [1]. It happened when a code snippet known as the "Brain" virus appeared on several dozen diskettes at the University of Delaware. Over the past two decades, the number of viruses has been increasing rapidly. We have seen several attacks that caused great disruption to the Internet and brought huge damage to organizations and individuals. For example, the Code Red worm outbreak in 2001 affected systems running Windows NT and Windows 2000 server and caused damage in excess of $2 billion [2]. Computer virus attacks will continue to pose a serious security threat to every computer user. Today viruses afflict at least a million computers every year. Users spend several hundred million dollars annually on antivirus products and services, and this figure is growing rapidly. In recent years, the number of viruses unleashed has increased dramatically and the extent of the damage (in lost data, time and productivity) is estimated to be several billions of US dollars per year [3]. People use a variety of methods to prevent, detect, and eliminate computer viruses to safeguard their computer information systems

[4, 5]. Unfortunately, the popularity of unknown and sophisticated virus likes metamorphic makes analysis and defense increasingly difficult for these traditional anti-virus methods. Therefore, new approaches must be found to effectively detect or identify the unknown viruses. Attacks by different kinds of viruses lead to different types of problems. Lots of effort has been done to minimize these attacks, but as time goes on, it becomes an obvious fact that the attacking technology is always getting one step ahead than the preventing technology. However, current analysis detection may not suitable to detect the evolved of today's virus. In this paper, we conduct an analysis that metamorphic viruses using obfuscation technique can bypass any signature-based detection. The rest of the paper is organized as follows: Section 2 describes the proposed detection approach and experiment setup. Section 3 discusses all the findings, analysis and conclusion that have been obtained from the experiment. Finally, the conclusion of this work is described in Section 4.

2. Proposed Detection Approach and Experiment Setup

As metamorphic viruses employ complicated techniques, many different methods have been developed to detect metamorphic viruses. Each detection method has its own pros and cons. Some of the detection techniques described in Symantec's white paper [6, 7, 8, 9] are highlighted below.

Geometric detection technique relies on "shape heuristic"; this allows to find whether a file is infected, or not, by learning the file structure of the virus and looking for learnt structures in the infected files. Often, this technique is prone to false positives as it simply learns the layout of the virus and does not learn about the virus at the instruction level.

Code emulation is employed by creating a virtual machine which emulates the underlying hardware including processor, memory, and peripherals and runs an operating system. This technique detects viruses by running suspicious files on its guest virtual machine and looks for any malicious activities and patterns. The above technique has the ability to detect complicated viruses but it needs considerable system resources to create a virtual machine.

Metamorphic viruses form families of viruses. Even though members in the same family mutate and change their appearances, some similarities must exist for the variants to maintain the same functionality. Detecting virus variants thus reduces to finding ways to detect these similarities. As our research is focused on using signature based for metamorphic virus detection. From here we will study the efficiency the obfuscation technique applied in the viruses' seed to evade from being detected. To run this analysis, we will use signature based detection as our method and perform the effectiveness of signature based detection in detecting those kinds of metamorphic variants.

2.1 Signature Based as Virus Detection Tool

Signature Detection is the oldest and most popular virus detection technique used today. Each virus is searched for a string of bytes that is unique to it, which becomes the signature of the virus. Signatures, also called "Scan Strings," sometimes depend upon the placement in the virus code. Scanners use a signature collection to identify known viruses and are almost certain to detect them. By constantly increasing virus's collection, signature scanning should be effective and efficient. Signatures can also contain wild cards that allow a few bytes to be anything. In the case of register swap-ping, the signature differs by the few bytes that contain the registers, but the other bytes remain same. In such cases wild cards are beneficial in

identifying new strains with an old signature. Signature extraction is a challenge in itself; a small signature would match to other non-virus programs and a long signature would be over fitting and may not identify new strains. To overcome this, multiple scanners are used on the same system. These scanners use a different set of signatures and help in identifying whatever signatures one scanner misses. As being discussed, viruses are clever at changing their look with alternating source code. A good mutation engine will generate very different strains and each strain will not have the signature of the original virus. In the case of metamorphic viruses, it is not possible to have a unique signature for the virus family. This means that although signatures of various strains are known there is always a good chance that another strain will succeed in bypassing the signature detection. Therefore, for this research we will conduct an analysis on the effectiveness of signature based in detecting metamorphic viruses.

2.2 Signature Based as Virus Detection Tool

In Section 1, we have emphasized on the main purpose of this paper, which is to ana-lyze the common methods of viruses attack, techniques applied to avoid detection and how technologies implemented in AV programs to detect the viruses prevalence. In order to accomplish this target, a step by step approach will be discussed more in this chapter, starting from collecting data and creating virus seed as a sample for the experiment. Then we will proceed in processing and analyzing all the inputs, and up until producing valuable outputs from the analysis.

1. Testing Phase. The experiments in this paper consisted of four major phases. The first phase involved creating the seed virus required for this paper and running baseline checks on the seed virus using pattern based detectors. Phase two involved running our code obfuscation technique on the seed virus to generate a family of metamorphic variants of the seed virus. The final phase involved testing the metamorphic viruses created by our engine using signature based detectors and similarity test.

2. Creation of the Seed Virus. The virus generator used for creating the seed virus for this paper was the Next Generation Virus Creation Kit (NGVCK) from vxheav-ens.com. For this experiment the viruses we created were unencrypted. The NGVCK virus creator generated the assembly language code for the virus which we assembled using TASM assembler and converted into an executable file.

3. Creation of Metamorphic Variants. After making certain that the seed virus was detected by a pattern based scanner it was run through our code morphing engine to create metamorphic versions. For the purposes of our experiment 20 variants of the seed virus were created. The code morphing engine reads the assembly code for the virus divides the code into blocks and then randomly shuffles the block order while simultaneously inserting some dead code blocks. The detailed process of applying obfuscation technique will be described below. Firstly, care was taken while splitting the code into blocks to make sure that none of the blocks ended with a jump instruction or a NOP instruction. After the blocks were created, the starting address of each block was stored in an array and a conditional jump instruction pointing to the next block was added at the end of each block. Next, some small blocks of dead code were added at the end of the array storing the generated code blocks. In the next step the small blocks of codes were randomly shuffled, in other words the logical order was changed and the results were stored in a text file.

The above process was repeated multiple times (20 times in the case) and the result is stored in different files. Care was taken while changing the logical order of the block to ensure that the first block was the same as that in the original code. According to Borello and Ludovic [10], all metamorphic viruses created by this engine always have the same entry and exit points (blocks).

Hence the virus was not parsed once it has reached the end of the last block. Though the blocks were linked using the conditional jumps, the original logical sequence could not be achieved unless the first block was parsed first. Figure 1 shows a side by side comparison of two variants created by our code obfuscation engine (VIRUS1.asm and VIRUS2.asm) and illustrates the difference in code between the metamorphic variants. We can see the labels and the jump instructions inserted between the blocks and the differences in the block order. It is also evident that we keep the starting block in the same place.

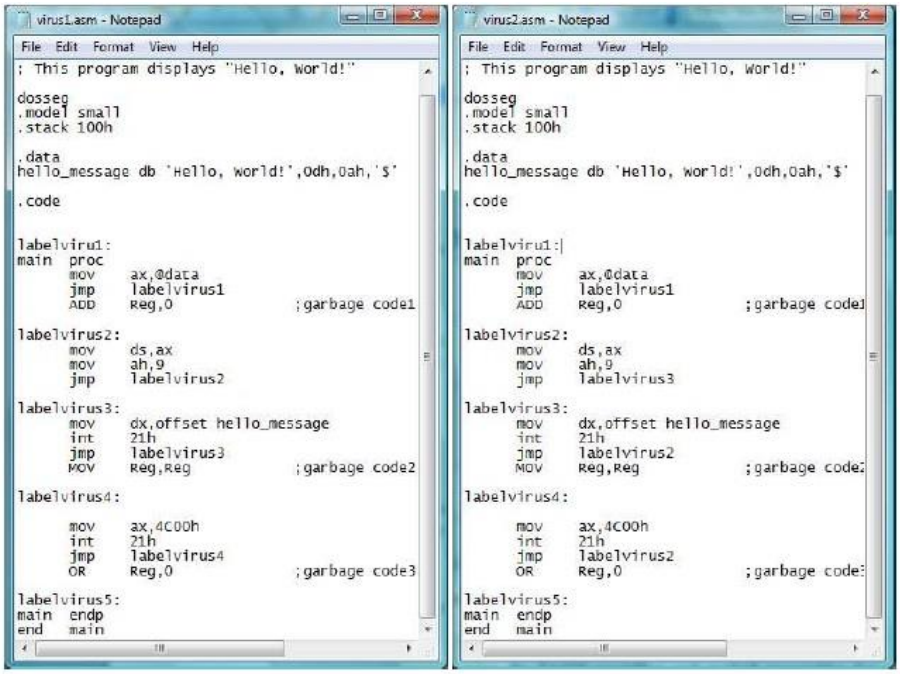

Figure 1. Two metamorphic variants generated by our code morphing engine

After creating the metamorphic variants of the original virus we assembled and linked these variants using the TASM assembler and created executables for each of them.

Testing Metamorphic Variants with Commercial Virus Scanners. In the third phase of our experiments the metamorphic viruses created in the second phase were tested with a signature based antivirus scanner. First, all the metamorphic viruses were scanned using the same scanner which is Kaspersky Internet Security 2010. It will use for checking the prevalence of the seed virus and its variants generation. Before the tests were conducted, we will update the antivirus so it has the latest of database signatures of any existing virus.

Metamorphic Copies Similarity Test. Firstly, similarity test were conducted to compares and reports the percentage of similarity of two assembly programs. From the similarity test we can measure the code diversity of the morphed copies. To compute the similarity between two files, we followed the following steps [11]:

1. Given two assembly files a.asm and b.asm, extract opcode sequences from each file excluding comments, blank lines, labels, and other directives. Let's call these opcode sequences *A* and *B* for the files a.asm and b.asm respectively

2. Consider m and n are the number of opcodes in *A* and *B* respectively.
3. Each opcode in *A* and *B* is assigned a number in ascending order i.e. first opcode is assigned 0, second opcode is assigned 1, third opcode is assigned 2, and so on.
4. Opcode sequences of *A* and *B* are divided into subsequences of length 3.
5. Every subsequence in *A* is compared with all subsequences in *B*. It is considered a match if the opcodes of any subsequence in *A* is same as the opcodes of any subsequence in *B*. These opcodes can be in any order. For example *A* is (mov,call,sub,add,test) and *B* is (mov,test,add,call,sub). The sequence (call,sub,add) in *A* matches with (add,call,sub) of *B*.
6. All such matches of *A* are computed and added together to find total number of match. This total number of matches is divided by m to get the similarity percentage of *A* (*X*).
7. Similarly the similarity percentage of *B* (*Y*) is computed.
8. The average of *X* and *Y* will give the actual similarity percentage between files a.asm and b.asm.

We have provided 4 generations of metamorphic copies for the test and compared with the virus seed. These comparisons were performed using the default settings of similarity test, 5 opcodes in a sequence is considered a match. Each of the generations will consist 5 different of virus variants. Each type of variant is different from others in re-ordering of the blocks but have the same length of opcodes. Its means each of the variants in the same generation will have the same size. This similarity test consists of two test cases:

1. Similarity of the 4 different virus generations
2. Similarity of two variants in N same generations

Virus Profile for the Existing Metamorphic Virus. For this paper, we have gathered the information on the existing metamorphic virus. The information will describe the characteristics belonging to the viruses. All the information is an analysis representing the extent to which something exhibits various characteristics.

3. Results and Discussion

Generally, this chapter will discuss on the results and findings from the study that was conducted. The first section consists of the outcome from the tested phase against the signature based detector. In this part, we will view the similarities between the viruses from the same metamorphic family that was trained on. The next section will be compiling on the findings of the existing metamorphic viruses. Lastly in the final part, the vulnerabilities existed in variant of metamorphic viruses will be discuses and presented.

Detecting Metamorphic Variants with Commercial Virus Scanner. After different variants were created from the virus seed, we have conducted a testing with the commercial antivirus scanner (signature based detection). First, all the metamorphic viruses were scanned using the same scanner which is Kaspersky Internet Security 2010. It will use for checking the prevalence seed virus and its variants and it failed to detect the presence of any virus. Figure 2 shows a screenshot capture of Kaspersky Internet Security 2010 after it was run on the folder containing the metamorphic virus executable generated by our code obfuscation engine.

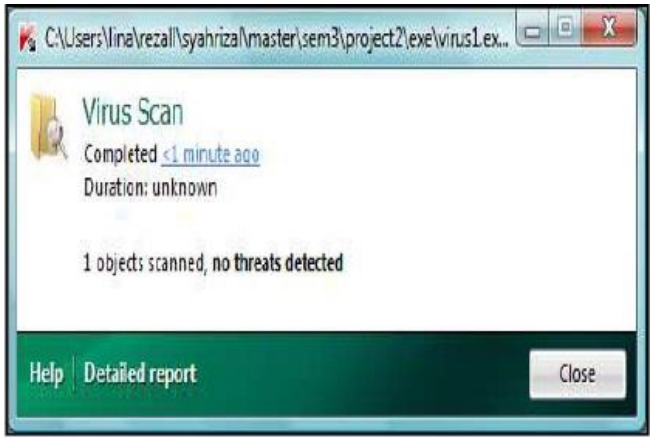

Figure 2. Kaspersky Internet Security 2010 fails to detect our metamorphic virus

Every time, obfuscation technique applied in the metamorphic virus variants it will create a new look to the variants. Metamorphic viruses change the appearance of the code in each generation. The size of file and the length of the opcodes are totally change but it still remains the same functionality of the virus, and that is the idea of metamorphism. Figure 3 will describe the idea of each generation of metamorphic virus look alike. It presents the different signatures of metamorphic viruses of each generation. As we know, signature based scanning involves a set of signatures, or a string of bytes which are found in viruses but not found in non-malicious code. Virus signatures are organized into databases. To identify virus infection, virus scanners check specific areas in files or system areas and match them against known signatures in databases. This kind of detection requires research on known viruses, and patterns within each virus need to be studied so that these signatures can be found. As being discussed above, new generation of metamorphic will look different from one another and this is the disadvantages to the signature based detection. It may produce false positive by the antivirus scanner in detecting metamorphic virus because of the well keep changing signatures of the viruses in each generation.

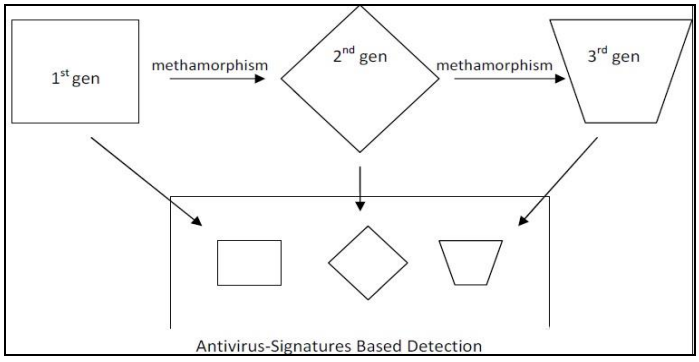

Figure 3. Different signatures of metamorphic virus generation

Here, we can conclude the effectiveness of obfuscation technique applied in each of the metamorphic viruses. They efficiency have evade from being detected by the antivirus scanners. Since the antivirus scanner is based on pattern match, signatures usually do not work that against the attacks with modifying behavior and obfuscation of the codes itself.

Metamorphic Variants Code Analysis. We have provided 4 generations of metamorphic copies for the similarity test and compared with the virus seed. These comparisons were performed using the default settings of similarity test, 5 opcodes in a sequence is considered a match. The result of this test is shown below in Figure 3. The similarity between the virus seed and 1st generation virus is about 60 percent. The similarity decreases with higher generations. 4th generation virus is about 20 percent similar to the virus seed. Clearly, similarities between 4th generation virus variants are lower than those between other generations. Compare with virus seed, which have an average similarity of 19.7 percent, we can see that the 4th generations viruses are largely different from virus seed, while the other generations virus variants are more similar to the virus seed. If the trends of the similarity continue, the new generation of metamorphic virus will lose their parent identity because of the decrease of similarity with the virus seed.

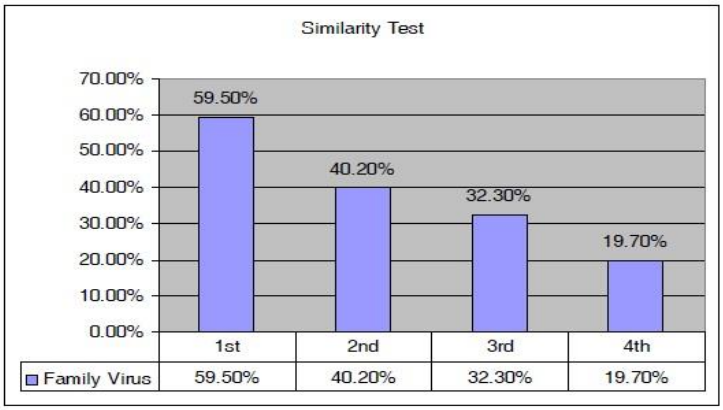

Figure 4. Similarity results of the 4 different generations

After conducted similarity test, we have found that each time the metamorphic engine applied to the seed virus, the number of opcode increase in each generation. The dissimilar length of the compared files may influence the similarity test. So we compared a pair of viruses from the same generation. The viruses from the same generation are of similar length 1st generation viruses are about 51 percent similar whereas 4th generation viruses are about 7 percent similar as shown in Figure 5. If we take a look at the graphs for the pair of 1st generation viruses, we can see that the real matches are higher with 50.5 percent similarities between them. This indicates that virus variants of the 1st generation virus have identical opcodes at identical positions. This is obviously not very effective metamorphism. On the other hand, the matches between the 2nd and 3rd generation virus pair are slightly lower with similarities between 25.7 percent and 12.4 percent. This shows that identical opcodes appear in different positions of the two virus variants in each generation. From this evidence, we can say that 4th generation (6.7 percent) virus variants have a greater morphing ability than the other 3 generations. 4th generation virus is the most effective in the sense that the match segments are very small and that they are way off the similarity.

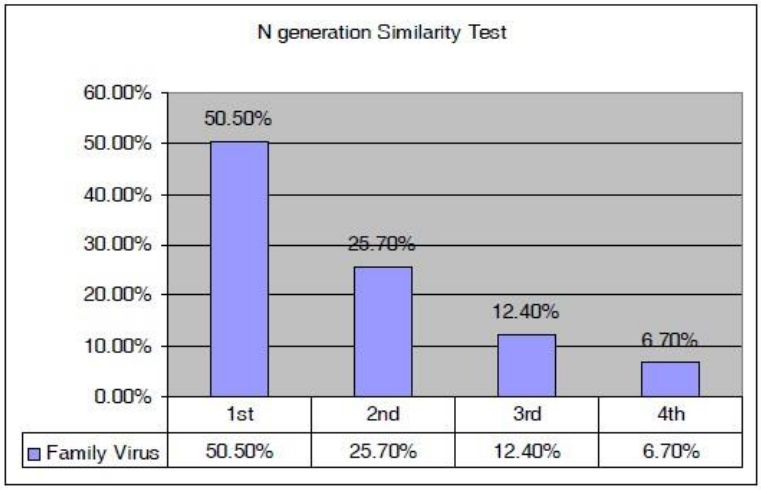

Figure 5. Graph of Similarity of two N generations

In our metamorphic engine, we employed code obfuscation techniques such as jump statement insertion, dead code insertion, and block re-ordering.The technique being applied have made each of the metamorphic variants generation are different in physically in size, length and sequence of opcodes. The similarity test compares two assembly programs and calculates the percentage of similarity between them. Specifically the generational variants of the same metamorphic virus family despite their differences do share a high degree of similarity especially when compared to virus seed because they tend to differ a lot from virus seed. This is obviously very effective metamorphism have been applied each time of code obfuscation techniques being employed.

Existing Metamorphic Virus Profile. For this section, we have compiled all the information on the existing metamorphic virus. The information will describe the characteristics belonging to the viruses. All the information is an analysis representing the extent to which something exhibits various characteristics.

Metamorphic Viruses Disadvantages. To avoid being detected by emulation, some viruses rewrite themselves completely each time they are to infect new executables. Viruses that use this technique are said to be metamorphic. To enable metamorphism, a metamorphic engine is needed. A metamorphic virus is usually very large and complex. For example, W32/Simile consisted of over 14000 lines of assembly code, 90% of it part of the metamorphic engine. Metamorphic viruses writer are step forward from the anti-virus scanners company [12]. Anti-virus scanners faced a lot of limits in static and dynamic analysis technique. Below are several techniques that used to avoid from the anti- virus scanner from detected their existence.

1. Attacks on disassembly
2. Attacks on procedure abstraction
3. Attacks on control flow graph generation
4. Attacks on data flow analysis
5. Attacks on property verification

In detecting virus, Lakhotia *et al.*, [13] believed that antivirus scanner must applied the same techniques that metamorphic virus creators use which spot the weaknesses of the metamorphic

viruses. It also explained the theoretical limits facing by virus creators are an advantage to the antivirus researchers in their effort to challenge the existing of the virus. AV scanner technology is constrained by the theoretical limits of program analysis techniques. A metamorphic virus is a manifestation of these limits. It turns out that to enjoy its advantage, a metamorphic virus too depends on program analysis techniques, because in order to mutate, a metamorphic virus must analyze its own code. Thus a metamorphic virus cannot use tricks that will fool its own analyzer. This handicap of a metamorphic virus can potentially be exploited to develop AV scanners. The Achilles' heel of a metamorphic virus, as described by Choucane and Lakhotia [14], is its need to analyse its own code. Virus researchers should be applied the same method the virus uses to analyse itself. Thus, the next step that should be taken by them is applied to all the transformation rules of the virus in reverse to reveal the real virus code. In order to achieve this goal, virus researchers must first have a sample of the virus in order to extract its transformation rules, assumptions, and algorithms.

4. Conclusion

This study consists of three different parts from the beginning. The first part is developed a code obfuscation engine as state and used this engine to create metamorphic variants of a seed virus and performed the validity about metamorphic viruses and signature based detectors [10]. In the next phase we validated another theory advanced, that methods based on Hidden Markov Model (HMM) can detect metamorphic viruses [15]. In other words, we want show that a collection of metamorphic viruses which are undetectable via signature detection techniques. Last but not least, we have developed a profile which contents the information about the existing metamorphic viruses' infection. The technique that used in the metamorphic to avoid from being detected is the obfuscation code. Metamorphic engine uses code obfuscation techniques to produce morphed copies of an original program. Generally the obfuscated code is more difficult to read and understand according to M. Stamp in his paper [16]. The same ideas also shared with Bordello and Ludovic [10] who explained the obfuscation technique that embedded in metamorphic viruses. Virus writers and anti-virus researchers generally agree that metamorphism is the way to generate undetectable viruses. To see how effective these code morphing engines are, and how much difference exists between variants of a given virus, we measured the similarity between virus variants generated. Our results show that the effectiveness of these codes morphing engine varies widely. While the 4th generation of the virus seed, share only a few percent of similarity with other virus generation and the similarity decreases with higher generations. The similarity between the virus seed and 1st generation virus is about 60 percent compares with 4th generation virus is about 20 percent similar to the virus seed. Furthermore, we have run the metamorphic virus variants under Kaspersky Internet Security 2010 to check the prevalence of the viruses. Unfortunately, it fails to detect the presence of any virus that generated by our code obfuscation engine. Although the signature based approach can effectively contain virus outbreaks in the right circumstances, metamorphic viruses, which obfuscate code and modify themselves as a method of disguise, so as to not match virus signatures in the dictionary.

Acknowledgements

This research is supported by Universiti Malaysia Pahang Research Grant (RDU120388)

References

[1] E. Al Daoud, I. H. Jebril and B. Zaqaibeh, "Computer Virus Strategies and Detection Methods", International Journal Open Problems Computer Science and Mathematics, **(2008)**.
[2] Washingtonpost.com Staff Writer, "A Short History of Computer Viruses and Attacks", **(2003)** February.
[3] S. R. Subramanya and N. Lakshminarasimhan, "Computer viruses", IEEE Potentials, **(2001)**.
[4] J. O. Kephart, D. M. Chess and S. R. White, "Computers and Epidemiology", IEEE Spectrum, **(1993)**.
[5] J. Kephart, G. Sorkin and D. Chess, "Fighting computer viruses", Scientific American, **(1997)**.
[6] P. Szor, "The Art of Computer Virus Research and Defense", Addison-Wesley, **(2005)**.
[7] P. Szor and P. Ferrie, "Hunting for Metamorphic", Symantec Security Response, **(2002)**.
[8] P. Szor, "Advanced code evolution techniques and computer virus generation toolkits", **(2005b)** March.
[9] P. Szor, "The Art of Computer Virus Defense and Research", Symantec Press, **(2005)**.
[10] J. -M. Borello and M. Ludovic, "Code Obfuscation Techniques for Metamorphic Viruses", Journal of Computer Virology, vol. 4, no. 3, **(2008)**, pp. 211–220.
[11] P. Mishra, "A taxonomy of software uniqueness transformations", Master thesis, San Jose State University, **(2003)** December.
[12] A. Lakhotia and M. Mohammed, "Imposing order on program statements to assist antivirus scanners", In Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04), **(2004)**, pp. 161-170.
[13] A. Lakhotia, A. Kapoor and E. U. Kumar, "Are metamorphic computer viruses really invisible? Part 1", Virus Bulletin, **(2004)**, pp. 5–7.
[14] M. R. Chouchane and A. Lakhotia, "Using engine signature to detect metamorphic malware", In Proceedings of the 4th ACM workshop on Recurring malcode (WORM '06), ACM Press, **(2006)**, pp. 73–78.
[15] W. Wong and M. Stamp, "Hunting for metamorphic engines", Journal in Computer Virology, vol. 2, no. 3, **(2006)**, pp. 211–229.
[16] M. Stamp, "Information Security: Principles and Practice", Wiley-Interscience, **(2005)** August.

Authors

Imran Edzereiq Kamarudin is currently lecture networking subjects in Universiti Malaysia Pahang, Malaysia. A certified CCNA and CCAI with 5 years working experience in network and telecommunications industry prior to lecturing His research area includes wireless network technology, mobile data network and wireless sensor network

Syahrizal Azmir Md Sharif, certified in CEH and ENSA, currently lecture security subjects in Universiti Malaysia Pahang, Malaysia His research area includes wireless network security and hacking method.

Tutut Herawan received a B.Ed degree in year 2002 and M.Sc degree in year 2006 degree in Mathematics from Universitas Ahmad Dahlan and Universitas Gadjah Mada Yogyakarta Indonesia, respectively. He obtained a PhD in Theoretical Data Mining from Universiti Tun Hussein Onn Malaysia in year 2010. Currently, he is a lecturer with Department of Mathematics Education, Universitas Ahmad Dahlan, Indonesia**.** He currently supervises four PhD and had successfully co-supervised two PhD students and published more than 120 papers in various international journals and conference proceedings. He has appointed as an editorial board member for IJDTA, TELKOMNIKA, IJNCAA, IJDCA and IJDIWC. He is also been appointed as a reviewer of several international journals such as Knowledge-Based Systems, Information Sciences, European Journal of Operational Research, Applied Mathematics Letters, and guest editor for several special issues of international journals. He has served as a program committee member and co-organizer for numerous international conferences/workshops including Soft Computing and Data Engineering (SCDE 2010-2011 at Korea, SCDE 2012 at Brazil), ADMTA 2012 Vietnam, DTA 2011-2012 at Korea, DICTAP 2012 at Thailand, ICDIPC 2012 at Lithuania, DEIS 2012 at Czech Republic, NDT 2012 at Bahrain, ICoCSIM 2012 at Indonesia, ICSDE'2013 at Malaysia, ICSECS 2013 at Malaysia, SCKDD 2013 at Vietnam and many more. His research area includes Knowledge Discovery in Databases, Educational Data Mining, Decision Support in Information System, Rough and Soft Set theory.