A Comparative Study of Metamorphic Malware Detection Techniques

Animesh Kumar Jha,[1] Abhishek Vaish,[1]
Simona Sternad Zabukovšek,[2] Samo Bobek[2]

[1] Indian Institute of Information Technology, Allahabad, India
icm2017001@iiita.ac.in, abhishek@iiita.ac.in
[2] University of Maribor, Faculty of Economics and Business, Maribor, Slovenia
simona.sternad@um.si, samo.bobek@um.si

Abstract Malware is a major threat in the evolving global cyber space. The different detection techniques that currently exist are insufficient at detecting metamorphic malware, as they can change the internal structure of their code, thus keeping the flow of the programme equivalent to the virus. Commercial antivirus software depends on signature detection algorithms to identify viruses, however, code obfuscation techniques can successfully circumvent these algorithms. The objective of this research is to analyse the various detection techniques of such metamorphic malware used over the years and to unearth the strengths, weaknesses and advance research directions possible in the field of the detection of metamorphic malware.

Keywords: metamorphic malware, detection techniques, comparative analysis, antivirus, algorithms

University of Maribor Press

DOI https://doi.org/10.18690/um.epf.5.2022.4
ISBN 978-961-286-600-6

1 Introduction

Malware is an evolving and ever-growing threat for global cyber space. The number of different types of malware detected each year is constantly increasing as is their ability to circumvent detection techniques. New forms of morphic malware are emerging that are capable of changing their signatures and evading detection by signature-based algorithms.

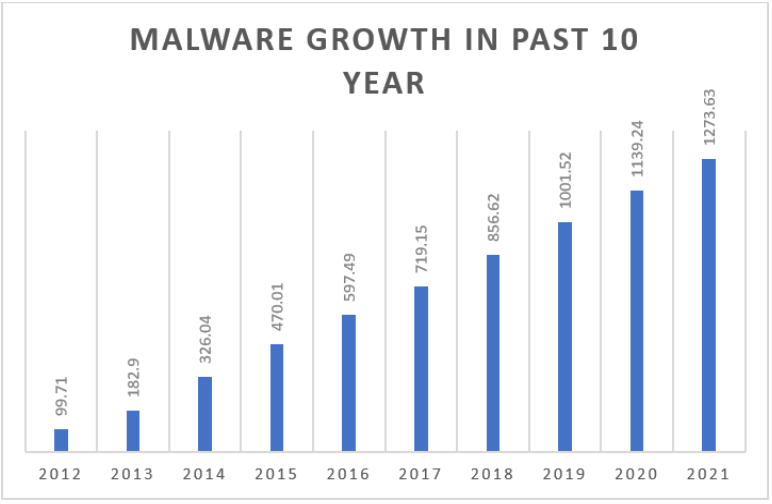

Figure 1: Overall development of new malware programmes over the last 10 years
Source: own.

Figure 1 highlights the constant growth in the production of malware over the past decade. While in 2020 the number of cases of malware reported were 1,273 million (Andreopoulos, 2021), the number of 'never-before-seen' types of malware were only 268,000 (Malware, 2021) and the total number of cases of new malware was around 146 million (Andreopoulos, 2021). Research by Camponi et al. (Campion, 2021) also reveals that over 66% of the types of malware are morphed from previously known threats. This highlights the fact that the maximum number of different types of malware in circulation are either old versions or morphed versions of existing ones. While there are techniques to easily detect the older variants using signature-based mechanisms, as well as some newer ones that use emulators or DFA-based techniques, the morphed variants are the most troublesome of all due to their 'evasive' nature.

Furthermore, research carried out by the Ponemon Institute (Ponemon Institute, 2018) into the state of endpoint security risk concluded that at least 76% of organisations are totally dependent on commercial antivirus programmes that use signature detection techniques to detect any intrusions or vulnerability. This, in turn, makes organisations vulnerable to metamorphic malware and makes a compelling argument for research and development in the detection of such malware. These types of malware have a mutation engine that takes in the code as input and returns a morphed version of the code in each iteration. The morphing is done on the basis of semantic preservation techniques, which ensures that while the code signature changes, while the effect of the code essentially remains the same. In other words, in metamorphic viruses, the physical appearance of the source code is morphed, while the logical flow remains the same. This in turn changes the hash of the signature code of the code, hence it is difficult to detect these metamorphic viruses using signature detection algorithms.

2 Background

Campion et al. (Campion, 2021) proposed an analysis on the schema and working of metamorphic malware by attempting to develop a framework that can produce the original variant of the malware by analysing different samples of the same malware and detecting the semantic preserving transformation rules applied to obtain the original malware code. This exercise also helps bring insights into the metamorphic malware engine or the mutation engine. The research reveals that, on average, over 90% of the code in the metamorphic malware is that of the engine itself. The engine is served the entire code as its input, and a morphed version is thrown as output, using various transformations, which essentially keeps the semantics the same. The engine also has a definite structure with various parts:

1. Disssembler - converts code to assembly instructions
2. Code Transformer - applies code obfuscation techniques
3. Assembler - converts mutated code to binary instructions

Various techniques have been found for the detection of malware. Some of the major ones are:

2.1 Signature Detection Techniques:

The most commonly used technique is signature-based algorithms, which is used by most of the commercially available antivirus software. Signature-based algorithms use the physical structure of malware to distinguish the type of malware. The algorithms use a database of malware signatures to detect potential malware. Metamorphic malware can change its physical structure while retaining the same control flow, thereby making it difficult to detect using signature-based techniques. Some other techniques for detection are:

2.2 Behavioural Detection:

This detection method uses the dynamic nature of the malware rather than the conventional static method of signature detection. The extraction of dynamic behaviour is carried out by executing a malware file in isolated surroundings (Kakisim, 2020). The main advantage of behavioural detection is when a computer virus looks similar to a benign programme but its functional aspect recognises it as harmful programme. In such cases, the above technique along with machine learning algorithms can be used to classify a record as harmful malware or a normal programme. Research by Desai and Stamp concludes that a metamorphic malware code can have an almost 93% similarity to a benign application, thus making it far more akin to benign files than malware (Desai, 2010).

2.3 Anomaly Detection Technique:

Anomaly detection chips away at the methodology of identifying if the document present follows a typical behavioural aspect. It beats the restrictions created by signature-based detection algorithms by utilising heuristic-based ways to deal with recognising ordinary behaviour. If a file is not classified as normal, then it is classified as harmful malware. In such cases, the notion of the anomalous and normal behaviour is expounded by the user, hence an accurate classification is not provided. A malware detection emulator has been created to distinguish the input programmes into different classes based on the appearance of the record. Once the classification is done then a review is made of whether it is harmful malware or an exceptional false positive case of being malware. In this research, the authors have attempted to morph malware codes and later create a classifier using machine learning algorithms

to detect these morphed malware files. They also go on to discuss various morphing techniques that can be employed to bypass detection and analyse the accuracy rate for such techniques. The subsequent sections deal with existing work and a literature review, followed by a proposed novel methodology for detecting metamorphic malware.

2.4 Newer Methodologies

Opcode Frequency: Research by Kakisim et. al. (Kakisim, 2020) revolves around generating a framework for the detection of metamorphic malware using automata principles. It establishes that every version of metamorphic malware is different from all the others, and the possible number of versions is so huge that a database cannot be maintained, thereby removing the possibility of using commercial anti-virus software or any pattern matching algorithms. Therefore, the authors suggested that the control flow of the programme is used rather than the code itself. They attempt to develop an opcode (instruction machine code) graph and then superimpose different versions of it to detect the similarities between different types of malware to find the engine code. They use the assumption that different variants formed of the same malware will have a common engine-specific pattern. The problem observed with this methodology was that it was necessary to have multiple samples of the same malware variants in order to be able to predict whether the malware had the same engine-specific variants. The possibility of detecting a totally new variant does not prevail in this research.

Hidden Markov Models: The use of Hidden Markov Models (HMM) for detecting malware has evolved as a popular research domain. This is due to the fact that any computer programme can be represented as a sequence of instructions, thereby treating a programme as a time series – an ideal condition for the use of HMMs (Stamp, 2004). Annachhatre et. al. (Annachhatre, 2015) applied HMMs and cluster analysis to detect unknown variants of malware by analysing the control flow of the programme. The HMM is trained on the basis of given observation sequences. The HMM is initially trained for a variety of compilers and generators of malware – training involves both, forward and backward algorithms and scoring is done by the forward algorithm. Finally, clustering is carried out using K-means.

Toderici et.al. (Toderici, 2013) showed that metamorphic malware can evade detection by HMM in the event that it uses morphing with instructions from benign files.

William B. Andreopoulos (Andreopoulos, 2021) used the assumption that a programme is basically a set of instructions that are executed in a sequence, therefore he proposed the use of sequence-based machine learning to derive insights to detect malware. The author also proposed a framework that would work in cases of morphic malware, i.e. polymorphic and metamorphic malware, and also used HMM principles and Long Short-Term Memory (LSTM) networks.

Genetic Algorithm: Javaheri et al. (Javaheri, 2021) proposed a novel approach to using a genetic algorithm for detecting future variants of targeted and metamorphic malware. The researchers extracted a sequence of system API calls using various filters. The authors unpacked and executed the files and opted for a sophisticated behavioural analysis to ascertain the classification of the file as harmful or benign. Dynamic unpacking is performed based on kernel-level memory dumping. Once unpacked, the malware features were extracted through parsing in order to find the relevant sequence flow to model the malware behaviour. The malware is further executed in a sandboxed environment and its activity is tracked and recorded. Finally, a genetic algorithm (GA) is employed, taking in each behaviour pattern as a chromosome and each system call mapped to a gene. Linear regression was used to model the formation of both behaviour and chromosomes.

Support Vector Machine: Devendra et. al. (Mahawer, 2014) presented a detection technique for metamorphic malware using machine learning techniques. The authors proposed the use of a support vector machine (SVM) as a tool to detect such variants of malware, using a specialised kernel for the model, called the histogram intersection kernel.

K. Kancherla et. al (Kancherla, 2013) also proposed the use of SVM for the classification of malware. They used the malware executable to be transformed into an image called the bytecode image, and then intensity-based and texture-based features are extracted to predict the code signature. SVM is then employed for bifurcation of dataset into benign and malware signatures. An accuracy of 95% is

reported in the paper. Code obfuscation and morphing was not incorporated, therefore limiting its applicability for the detection of metamorphic malware.

2.5 Comparative Analysis Based on Literature Survey

In this section, the authors of this study provide an analysis table of the various detection techniques for metamorphic malware.

Table 1: Comparative Study of Reviewed Papers

Sl.No.	Researchers	Technique	Approach	Advantages	Shortcomings	Dataset size
1	Elhadi et al. (2013)	API calls	An API call graph was developed by integrating an API call and system resources.	Accurate graph generation through sequence profiling data dependen-cies.	Lower efficiency on polymorphic malware. The time complexity of the API graph does not make it viable..	514
2	Javahe ri et al. (2021)	API calls and geneti c algorit hm	Unpacks files and studies the API calls made in a behavioural analysis to ascertain benign or corrupt files. Later, a genetic algorithm is applied with behaviour as a chromosome and a system call as the gene.	The behavioural approach allows the detection of never-before-seen malware. The API call sequencing allows the detection of various types of web-based content.	2 samples or more are required to generate variants and perform crossovers. Recent metamorphic malware can hide behaviour in a controlled environment.	NA
3	Campi on et al. (2021)	Base malwa re	The proposed algorithm assumes there is a base malware code.	Provides an-indepth insight into obfuscation techniques and helps in understanding the working of the mutation engine.	Only works if the base malware is recognised and cannot function for never-before-seen malware.	NA

Sl.No.	Researchers	Technique	Approach	Advantages	Shortcomings	Dataset size
4	Kancherla et. al (2013)	Bytecode Image and SVM	Converted malware executable into a bytecode image and then used SVM to classify	Reported accuracy of an average of 95%..	Uses a supervised algorithm and generic kernel function. Can malfunction in the event of high obfuscation in code. The dataset used is inaccurate.	NA
5	Agrawal et al. (2012)	CFG	Works to eliminate the graph comparison problem. Uses normalised edit distance technique.	Attempts control-flow analysis (semantic) that can detect newer viruses.	Fails against non-assembly malware. Tested on a very small dataset.	18
6	Eskandari and Hashmi (2011)	CFG + API Calls	Detects metamorphic malware by making use of a control-flow graph (CFG) and an API call graph.	Able to detect obfuscation in files.	Provides poor results with non-assembly malware. Computation speed is very slow.	4445
7	Martins et al. (2014)	Dependency Graph	Forms a dependency graph of an executable file. Based on this dependency graph, it identifies the closely related nodes.	Eliminates the impact with very high accuracy.	Identification on closely related nodes and events should be based on the statistical inference technique.	63
8	Todorici et.al. (2013)	HMM	Uses HMM in conjunction with a chi-square distance calculation.	Capable of detecting malware even if benign files are used to obfuscate code.	The method is not useful for real-time detection.	NA
9	Annachhatre et. al. (2015)	HMM and Cluster Analysis	HMMs and cluster analysis have been applied to detect known variants of malware by	HMM is trained on the basis of the given observation sequences.	Uses a straightforward scoring system on a forward algorithm. A more specific	NA

Sl.No.	Researchers	Technique	Approach	Advantages	Shortcomings	Dataset size
			analysing the control flow of the programme. Clustering is done based on KNN.	HMM is trained for a variety of producers and generators of malware.	scoring system can be produced to increase accuracy.	
10	Andreopoulos (2021)	HMM and LSTM	Assumes that a programme is essentially a sequential instruction set, therefore a sequence-based ML approach is provided.	Can provide real-time detection in conjunction with antivirus software. Potential to detect both polymorphic and metamorphic variants.	Uses simple kernel functions for HMMs, which affects accuracy. High false positives when code is obfuscated with benign samples.	NA
11	Van Nhuong et al. (2014)	Hybrid	Uses a hybrid method formed from two known methods. Developed a detection system for detecting malware, which consists of the advantages of both the base methods.	Removes malware obfuscation by extracting automation of semantic sets. Detection rate of almost 100%.	Very high processing time. No support for real time detection.	186
12	Mahawer and Nagaraju (2014)	Opcode	This method is derived from the most frequently occurring histograms of opcodes in disassembled files.	Very efficient at detecting the insertion of dead code, registry renaming and code reordering.	Unable to detect malware that can self-obfuscate, thus leading to a very high rate of FNs.	2121
13	Kakisim et. al. (2020)	Opcode	Generates an opcode graph for different iterations on the same malware then superimposes the image to find	Can effectively detect morphed versions of a malware family.	Based on the assumption of an identical engine for all interations. Multiple samples of the same malware	NA

Sl.No.	Researchers	Technique	Approach	Advantages	Shortcomings	Dataset size
			commonalities to deduce the malware engine.		are required to detect malware of the same family.	
14	Alam et al. (2015)	Opcode	This method first computes the weight of the mail pattern and control flow.	Does not depend on any platforms. Supports automated analysis by making use of an intermediate language.	Very high complexity for a large dataset. Opcode frequency could be affected if any optimisations are made in the compiler. Highly vulnerable to code obfuscation.	1251

3 Conclusion

The aim of this research was to learn about the detection of metamorphic malware and to analyse the existing techniques for such malware. The analysis shows that HMM, opcode-based analysis, and control flow graph-based analysis are the major advancing areas. The major issue spanning all areas in this regard is that to date there has been no testing and validation of these approaches on standard databases with ample data points for testing. Most of the research does not specify the testing criterion, and the ones that do, have an insufficient database to ensure its production readiness. There is a need to propose research in this domain using a larger dataset.

In addition, another very important factor that comes into play is the kind of obfuscation techniques that have been used to morph malware. The better the morphing, the closer it is to a real-life detector, and many of the works reviewed in this regard were not as dedicated to obfuscation as they were to detection. Based on this study, it can also be established that there is still future scope for research by instilling some specific approaches in the areas of HMM with different scoring systems, or by using a genetic algorithm or even by using the unpredictability of mutation to be detected through the use of fuzzy neural networks (FNN).

4 Future scope

Based on a review of the different methods for the detection of metamorphic malware, the authors of this study have identified some of the major research domains where research is ongoing:

1. Hidden Markov Model
2. Support Vector Machine
3. Genetic Algorithm
4. Emulator-based approach

All of the domains suffer from some kind of deficiency. A wholesome framework is required that can potentially detect never-before-seen variants of malware with lower complexity and higher efficacy.

To this end, machine learning algorithms could be a solution, however, existing studies into their use have not be very promising, other than in terms of SVM. Additionally, the detection of malware is predominantly a classification problem, which also uses the attributes of pattern recognition. The use case of a FNN also revolves around these two pillars, thereby making it a possible alternative approach.

In view of the lack of studies into FNNs for the detection of malware and its unsupervised nature, it could be a potential future research topic. Fuzzy neural networks have been widely used for pattern recognition, on which the opcode frequency histogram approach also relies. These networks have also been used in detecting variants of COVID, where COVID-like metamorphic malware is also an 'evasive' variant, i.e. it morphs into new forms. Similar learnings can also be applied to the base malware identification approach.

References

Agrawal, H., Bahler, L., Micallef, J., Snyder, S. and Virodov, A. (2012) 'Detection of global, metamorphic malware variants using control and data flow analysis', MILCOM 2012 - 2012 IEEE Military Communications Conference. doi: 10.1109/milcom.2012.6415581.

Alam, S., Horspool, R. N., Traore, I., & Sogukpinar, I. (2015). *A framework for metamorphic malware analysis and real-time detection. computers & security*, 48, 212-233.

Andreopoulos, W. B. (2021). *Malware detection with sequence-based machine learning and deep learning. In Malware Analysis Using Artificial Intelligence and Deep Learning* (pp. 53-70). Springer, Cham.

Annachhatre, C., Austin, T. H., & Stamp, M. (2015). *Hidden Markov models for malware classification. Journal of Computer Virology and Hacking Techniques*, 11(2), 59-73.

Campion, M., Dalla Preda, M., & Giacobazzi, R. (2021). *Learning metamorphic malware signatures from samples*. Journal of Computer Virology and Hacking Techniques, 1-17.

Desai P, Stamp M (2010), *A highly metamorphic virus generator*, Int. J. Multimedia Intelligence and Security, Vol. 1, No. 4, 2010

Elhadi, A. A., Maarof, M. A., & Barry, B. (2013). Improving the detection of malware behaviour using simplified data dependent API call graph. International Journal of Security and Its Applications, 7(5), 29–42. https://doi.org/10.14257/ijsia.2013.7.5.03

Eskandari, M. and Hashemi, S. (2011) 'Metamorphic malware detection using control flow graph mining', International Journal of Computer Science Network Security, Vol. 11, No. 12, pp.1-6.

Hum.-centric comput. inf. sci., December 2018. 8(1)

Irshad, M., Al-Khateeb, H. M., Mansour, A., Ashawa, M., & Hamisu, M. (2018). *Effective methods to detect metamorphic malware: a systematic review*. International Journal of Electronic Security and Digital Forensics, 10(2), 138-154.

Javaheri, D., Lalbakhsh, P., & Hosseinzadeh, M. (2021). *A Novel Method for Detecting Future Generations of Targeted and Metamorphic Malware Based on Genetic Algorithm*. IEEE Access, 9, 69951-69970.

Kakisim, A. G., Nar, M., & Sogukpinar, I. (2020). *Metamorphic malware identification using engine-specific patterns based on co-opcode graphs*. Computer Standards & Interfaces, 71, 103443.

Kancherla, K., & Mukkamala, S. (2013, April). *Image visualization based malware detection*. In 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS) (pp. 40-44). IEEE.

Kienzle, D. M., & Elder, M. C. (2003, October). *Recent worms: a survey and trends*. In Proceedings of the 2003 ACM workshop on Rapid Malcode (pp. 1-10).

Mahawer, D. K., & Nagaraju, A. (2014). *Metamorphic malware detection using base malware identification approach*. Security and Communication Networks, 7(11), 1719-1733.

Malware. https://www.av-test.org/en/statistics/malware/. Accessed: 2021-11-25.

Martins, G.B., de Freitas, R. and Souto, E. (2014) 'Virtual structures and heterogeneous nodes in dependency graphs for detecting metamorphic malware', 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC), DOI: 10.1109/pccc.2014.7017069

Musale, M., Austin, T. H., & Stamp, M. (2015). *Hunting for metamorphic JavaScript malware*. Journal of Computer Virology and Hacking Techniques, 11(2), 89-102.

Rezaei, F., Hamedi-Hamzehkolaie, M., Rezaei, S., & Payandeh, A. (2014, September). *Metamorphic viruses detection by hidden Markov models*. In 7'th International Symposium on Telecommunications (IST'2014) (pp. 821-826). IEEE.

Sam Cook Data journalist, privacy advocate and Cord-Cutting Expert. *Malware statistics in 2021: Frequency, impact, cost & more*. https://www.comparitech.com/antivirus/malware-statistics-facts/, August 2019. Accessed: 2021-11-25.

Souri, A., & Hosseini, R. (2018). *A state-of-the-art survey of malware detection approaches using data mining techniques*. Human-centric Computing and Information Sciences, 8(1), 1-22.

Stamp, M. (2004). *A revealing introduction to hidden Markov models*. Department of Computer Science San Jose State University, 26-56.

Szor, P. (2005). *The Art of Computer Virus Research and Defense: ART COMP VIRUS RES DEFENSE _p1*. Pearson Education.

The 2018 state of endpoint security risk: Ponemon institute. https://www.ponemon.org/news-updates/ news-press-releases/news/the-2018-state-of-endpoint-security-risk.html. Accessed: 2021-11-25.

Toderici, A. H., & Stamp, M. (2013). Chi-squared distance and metamorphic virus detection. Journal of Computer Virology and Hacking Techniques, 9(1), 1-14.

Van Nhuong, N., Yen Nhi, V. T., Cam, N. T., Phu, M. X., & Dang Tan, C. (2014). SSSM-semantic set and string matching based malware detection. *The 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. https://doi.org/10.1109/cisda.2014.7035642

Wilding, E. (Nov 1990) Virus bulletin.