Effective Methods to Detect Metamorphic Malware: A Systematic Review

Mustafa Irshad, Haider M. al-Khateeb*, Ali Mansour*, Moses Ashawa, and Muhammad Hamisu

School of Computer Science and Technology
University of Bedfordshire
University Square, Luton, Bedfordshire, LU1 3JU, UK.

Email: mustafa.irshad@beds.ac.uk
Email: haider.alkhateeb@beds.ac.uk
Email: ali.mansour@beds.ac.uk
Email: moses.ashawa@study.beds.ac.uk
Email: muhammad.hamisu@study.beds.ac.uk
*Corresponding authors

Abstract – The succeeding code for metamorphic Malware is routinely rewritten to remain stealthy and undetected within infected environments. This characteristic is maintained by means of encryption and decryption methods, obfuscation through garbage code insertion, code transformation and registry modification which makes detection very challenging. The main objective of this study is to contribute an evidence-based narrative demonstrating the effectiveness of recent proposals. Sixteen primary studies were included in this analysis based on a pre-defined protocol. The majority of the reviewed detection methods used Opcode, Control Flow Graph (CFG) and API Call Graph. Key challenges facing the detection of metamorphic malware include code obfuscation, lack of dynamic capabilities to analyse code and application difficulty. Methods were further analysed on the basis of their approach, limitation, empirical evidence and key parameters such as dataset, Detection Rate (DR) and False Positive Rate (FPR).

Keywords – Metaphoric malware; Malware Detection; Review; Opcode; Control Flow Graph; API Call Graph

Biographical notes: Mustafa Irshad is a researcher in cyber security, he has MSc in Computer Networking from the University of Bedfordshire, UK.

Haider M. al-Khateeb specialises in Cyber Security and Digital Forensics. He holds a first-class BSc (Honours) in Computer Science and PhD in Cyber Security. He is a university lecturer, researcher, trainer and a Fellow of the Higher Education Academy (FHEA), UK. He is a lecturer in the School of Computer Science and Technology and conducts research within the Institute for Research in Applicable Computing (IRAC), University of

M. Irshad, H. al-Khateeb, A. Mansour, A. Ashawa, and M. Hamisu, "Effective methods to detect metamorphic malware: A systematic review", *International Journal of Electronic Security and Digital Forensics*, vol. 10, no. 2, pp. 138–154, 2018, ISSN: 1751-9128. DOI: 10.1504/IJESDF.2018.090948. [Online]. Available: https://doi.org/10.1504/IJESDF.2018.090948.

Bedfordshire. He is also an associate lecturer for QA Ltd. He supervises students at MSc/MPhil and PhD levels at these institutes.

Ali Mansour is a Senior Lecturer in Computing and Information Systems at the University of Bedfordshire. He has a BSc (Honours) in Computer Studies from Sheffield City Polytechnic (now Sheffield Hallam University) in 1987, and PhD in Computer Science from the University of Sheffield in 1990. He is a Fellow of the British Computer Society (FBCS), an Engineering Council Chartered Engineer (CEng), BCS Chartered IT Professional (CITP) and a Fellow of the Higher Education Academy (FHEA). His research areas are in Cyber Security, Networking, e-learning, and Medical informatics.

Moses Ashawa is a System Analyst at the Federal Ministry of Communication and Technology, Nigeria. He has a BSc (Honours) in Computer Science from Benue State University Makurdi, Nigeria, and MSc in Computer Security and Forensics (Distinction) from the University of Bedfordshire, UK. His keen interest is in Digital Forensics and Cyber Security.

Muhammad Hamisu hold a Bachelor of Engineering degree (B.Engr.) in Computer Engineering from Bayero University Kano, Nigeria in 2012, and an MSc in Computer Security and Digital Forensics (Distinction) from University of Bedfordshire, UK.

I. INTRODUCTION

The habit of downloading unknown files over the Internet seem to be the key reason for infecting computer systems with Malware. Malicious software has the potential to infect system files, write extreme data on adjacent memory, leading to buffer overflow, and change codes of other executable files. History of Malware evolution shows that many malicious software were written for fun or testing software behaviour. However, state-of-the-art malware is also developed for financial gain (Alam et al., 2014c), political influence, enabling anti-social behaviour such as cyberstalking (al-Khateeb et al., 2016), or to sabotaging the defence systems of a country. Consequently, malware coding became extremely sophisticated. Figure 1 demonstrates a simplistic view of malware evolution since the 1970s (Rad et al., 2012).

Figure 1. Malware evolution

Since the first malware written in the last century, a rivalry has started between defenders (e.g. researchers) and malware developers who continue to adopt new evasion techniques (Sharma and Sahay, 2014).

A first-generation malware is relatively easy to identify using traditional antivirus software. Detection in this case requires a library of static byte-signatures, hash-signatures and any code or strings depending on the technique used. Signatures of malicious files are stored in database files (.mdb, .idb, etc.) to facilitate the detection process. Infected files

are usually quarantined and could also be moved to a Sandbox for further analysis. Signature-based scanning is considered fast compared to non-signature-based detection. In contrast, malware writers nowadays use obfuscation techniques to evade detection (Karim et al., 2005). These techniques can be classified into pattern-based, content-based and protocol based methods. Their functionality includes the insertion of garbage code, changing the execution order of instruction, automatically regenerated code (through a metamorphic engine) and variable renaming (Christodorescu and Jha, 2004). This mutation capability of code is called 'metamorphism' in which the program changes the structural properties of the executable without any change in behaviour. Metamorphic malware is therefore a software program with the capability to change its code and signature on its propagation. It has its own encrypted virus-body coated with a decryption routine, both of which are changed in the propagation phase. Therefore, current antivirus programs have difficulties detecting this type of malware (Lyda and Hamrock, 2007).

Nonetheless, various methods, frameworks and detection systems have been proposed to encounter obfuscation techniques with the majority utilising Opcode Sequence, Control Flow Graph, N-gram and API Calls (as shown in Table 1). While every approach has its own benefits and limitation, a hybrid method combining two or more of these could overcome some of the challenges and increase performance. These methods perform static analysis prior to any file execution. Then, the process extracts Opcode and Bytecode sequences, and generate graphs by disassembling the file (Egele et al., 2012). Static analysis is considered robust yet suffers when the source code of the file is not available. Further, it does not have the ability to identify new malware (Wu et al., 2011). On the other hand, dynamic analysis does not rely on the code. Instead, it monitors the behaviour of the file during its execution in a Sandbox, which is a securely restricted environment to facilitate the analysis of unknown or untrusted files (Egele et al., 2012). Behaviour is more unique and consistent compared to the static features of a malicious file, it could therefore help to thwart the different variants of Malware and declare their obfuscation techniques irrelevant (Dai and Kuo, 2007). However, behavioural analysis is relatively more resource consuming, lengthy and could result in many false-positives (Fukushima et al., 2010). Hence, an integrated system balancing the trade-off between instant detection and long-term detection procedures could lead to effective and efficient solutions for metamorphic malware (Shijo and Salim, 2015).

The remaining parts of this paper are organised as follows: Section II shares the methodology and protocol used for this systematic literature review. Section III contains detailed critical analysis of the detection techniques surveyed. Finally, conclusions and further discussions are stated in Section IV.

II. METHODOLOGY

A systematic review is conducted with reference to the comprehensive guidelines published by Kitchenham and Charters, (2007). We aim to implement unbiased protocol to include recent studies on metamorphic malware detection. The following sections elaborates further details on the research questions, data selection and extraction strategy.

A. Research questions (RQs)

RQs. What type of detection techniques and methods have recently been presented and tested for metamorphic malware detection? Which of these techniques is more effective in

detecting metamorphic malware? And what are the key challenges facing the detection process?

We address these RQs with a critical analysis of the included studies.

B. Inclusion and exclusion criteria

Papers matching the following conditions will be included in this study:

- ✓ A recent study published within the last 5 years
- ✓ Papers must present a novel malware detection technique
- ✓ Papers must present empirical evidence e.g. statistics for evaluation purposes

Papers matching the following conditions will then excluded:

- ✘ Non-peer reviewed papers such as descriptive reports and technical reports
- ✘ Abstracts, editorials, and books.
- ✘ Papers not written in English.
- ✘ Papers focusing on malware *classification* methods.

C. Data Source and Search Strategy

Robust and popular databases covering the discipline of computer science and technology had been selected to be the data sources, these were: IEEE Xplore, ACM DL, Springer, Science Direct, and Scopus. Included studies have been published between January 2010 and June 2015. Logical operators 'AND' and 'OR', and Search keywords related to the proposed research question and topic were used; 'Metamorphic', 'Malware', 'Virus', 'Detection', 'Method', 'System' and 'Technique'. Targeted fields were the Title, Abstract, and Keywords. Advance search options were applied to enforce inclusion and restriction criteria such as publication time and article type. We have also examined the bibliographies of selected studies.

D. Primary Study Selection Process

The selection process was performed on the databases and reviewed by the authors accordingly. First, the retrieved number of filtered articles pulled from the databases was 113. These articles were then taken to the next process. Second, *Title and abstract base Selection*: the examination was scoped at these two parts for each article. When there was doubt, the article was included for the next phase. A total of 37 articles remained. And finally, *Full Text Selection*: Remaining articles were thoroughly read. A total of 21 articles were discarded, leaving 16 key articles most of which seem to have come from IEEE.

E. Data extraction

Data to be extracted from included studies will be: Article type (Journal, Conference, etc.); citation details; technique used; experimental results (e.g. Detection Rate (DR), False Positive Rate (FPR), and dataset size); strengths and limitations; and any technical challenges reported.

F. Evaluation measures

The effectiveness of the proposed malware detection methods can be measured by means of DR and FPR. Hence, it is also important to highlight the following:

- *True Positive (TP)*. The number of correctly identified malicious files as malware.
- *False Positive (FP)*. The number of incorrectly identified benign files as malware.
- *True Negative (TN)*: The number of correctly identified benign files as benign.
- *False Negative (FN)*: The number of incorrect identified malware files as benign.
- And *DR = TP/ (TP+FN)*, while *FPR = FP/ (TN+FP)* (Alam et al., 2014c).

F. Quality assessment threshold

A list of quality assessment questions facilitated the evaluation of primary studies in this systematic review. Studies must satisfy all the following threshold criteria with a 'yes':

1- Are the study objectives stated clearly?
2- Does the evaluation fulfil the purpose of the primary study?
3- Is the conclusion supported by empirical evidence?
4- Are the methods specified for data analysis and collection thoroughly described?
5- Were the DR or FPR reported?

III. Analysis of detection techniques

Included studies recovered various detection techniques, the sample have been analysed in Table 1, while main themes conceptualised in Figure 2 and discussed in the remaining subsections.

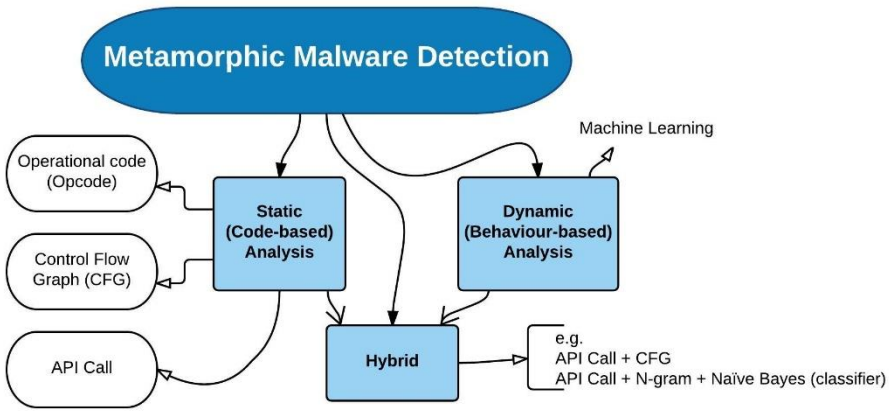

Figure 2 Metamorphic Malware Detection Approaches. References included in our review are discussed for each of these categories in the following subsections A to E.

A. Operational Code (Opcode)

An executable program is a construction of a series of machine language instructions. These instructions are composed of a pair; mostly a list of operands and operational code. Opcode is the portion of the code that specifies the operations while Operands (the data to be processed) could provide extra information about the executable files (Zolotukhin and

Hamalainen, 2014). The source code of a given software, including Opcode, is more consistent and therefore suitable to produce signatures for malicious software-classes compared to compiled code. However, processing overhead (Carrillo and Lipman, 1988; Santos et al., 2013) and compiler optimisation (Alam et al., 2014a) are examples of challenges to be addressed when Opcode detection is utilised. Further, Opcode distribution has weakness against obfuscation techniques (Mahawer and Nagaraju, 2013; Alam et al., 2014a), and it cannot be used to detect unknown malware (Rezaei et al., 2014a).

Vinod et al. (2012) used Opcode sequences to calculate the similarity among malware executable files. Three different types of signatures were reported. Multiple Sequence Alignment was used to create single and group signatures whereas probabilistic signatures were constructed using Pair-wise alignment. Rezaei et al. (2014a) used the same Opcode sequence technique to compare two pieces of code extracted from executable files with the help of the Mishra Method and a Hidden Markov Model. According to the proposed model, Opcode extracted from a file was compared by calculating the probability of an existing virus, and Edit Distance was used to calculate the distance (or dissimilarity) between two strings created by the executable's Opcodes with the help of probability factors and coordinate axis of a Detection Sphere. Rezaei et al. (2014b) amended the same approach to utilise Detection Circle instead of the previously used Detection Sphere.

Alam et al. (2014a) proposed a unique approach for metamorphic malware detection called SWOD-CFWeight (Sliding Windows of Differences and Control Flow Weight). They used an intermediate language MAIL (Malware Analysis Intermediate Language) (Alam et al., 2013) to transform assembly code which makes the detection process platform-independent. SWOD represented the differences among Opcode distribution, changes in size whereas CFWeight, monitored the controlled information flow of a program, this has helped real-time detection to some extent.

Mahawer and Nagaraju (2013) used Opcode with histogram intersection kernel and Support Vector Machine. Histogram intersection had been used to build an effective detection system for metamorphic malware. In this method, the detection method was based on Most Frequently Occurred (MFO) Opcodes in disassembled files, and code normalisation improved the detection rate. Opcode distance between malware and benign files was calculated with the help of the Euclidean Distance Equation.

B. Control Flow Graph (CFG)

CFG is a directed graph representation which has been in use for malware analysis and detection for many years. It is a representation of control flow of a program during its execution using graph notation such as nodes, straight line, edges. In CFG, statements such as assignments, copy statements, and branches are represented by nodes whereas control flow between every statement (i.e. what comes after next) is represented by an edge. CFG's performance is not affected by code modification (Paul and Mishra, 2014). However, subgraph isomorphism consumes a high amount of time for processing and has an NP-Complete and NP-Hard problem (Bai et al., 2009; Kim and Moon, 2010; Alam et al., 2014a). An ideal model of CFG runs under offline mode (Paul and Mishra, 2014).

Recent CFG proposals include (Agrawal et al., 2012), where the authors presented the use of Malware Abstraction Analysis (MAA) to compare high-level abstractions, these are flow graphs of targeted binaries that do not take low level syntax such as function call

and return, and conditional statements into consideration. Extracted graphs are referred to as 'Rooted Tree', and to compare these trees, the Edit Distance technique is used along with Approximating, Eliminating Search Algorithm (AESA). AESA helps to determine the nearest neighbour of the signature tree. Another proposal by Alam et al. (2014c) focused on real-time detection that is also platform independent. This was achieved by disassembling and translating the binary program into an intermediate language called Malware Analysis Intermediate Language (MAIL). Overall, CFG-based solutions can be very complex and expensive therefore may not be useful in filtering malware traffic in a high traffic link (Li et al., 2006).

C. API Call

Graphs are built by transforming the executable file into a call graph with the help of nodes and edges that respectively represent system calls and system call sequences. This technique utilises the relationship between the API Calls (Elhadi et al., 2013). Lee et al., (2010) developed a method based on semantic analysis in which call sequences were first converted to API Call Graphs. And to avoid the NP-complete problem, API Call Graphs were then reduced to code graphs. They were stored and used later to measure the similarity against code graph extracted from input program files. The method achieved a DR of 98% and was tested against three code obfuscation techniques, namely: code insertion, code reordering, and code replacement. Moreover, meaningless system call insertion is another obfuscation technique that could still increase the number of FPR, in response, Elhadi et al., (2014) developed Lee's method further by introducing sequence profile and a scheme for data dependency to generate more accurate call graphs. The idea is based on the observation that more than half of the discovered malware samples are derived from known samples. However, if polymorphic techniques are used to pack the malware, then extracted API call lists and its parameters would be incorrect.

Kwon and Lee (2012) method applied graph mining technique to construct semantic signatures. As graph matching leads to NP-complete, they used the matrix data structure technique for semantic graph matching by computing XOR operations only in contrary to (Lee et al., 2010). While in (Wu et al., 2013) call graphs were generated from a programme's function using assembly code adopting a breadth-first approach. Vertices were matched with the help of a function matching process and graph colouring techniques. Cosine similarity was used which could be problematic when similar functions or substitute instructions are used. A key phase of this method was to disassemble executable files, which is quite challenging, and the reliability of function matching based on graph colouring are reduced when instructions from one class are identified as equivalent to another e.g. the 'ECX' and 'MOV' instructions (Wu et al., 2013).

D. A Hybrid Approach

Malware with obfuscation techniques can deceive detection systems, it usually changes its code with pre-defined impact on its key behaviour. Therefore, signature-based detection becomes irrelevant while, as discussed earlier, behavioural analysis introduces time cost and a requirement for extra processing. Further, DR could arguably be enhanced when two methods are combined. To inherit advantages from multiple methods into an integrated solution, hybrid techniques were introduced. For instance, Eskandari and Hashemi (2011)

combined API Call Graph and CFG. The proposed system consisted of three phases. First, the executable is disassembled with a pre-processing algorithm to remove unnecessary statements and to generate CFGs. Then, API Call Graphs can be generated with the help of a labelling algorithm. Finally, features are generated based on the API Call Graphs.

A semantic set method was proposed by Van Nhuong et al. (2014a), it combines data mining techniques namely N-gram with a modified Naïve Bayes classifying algorithm to reduce processing time and increase accuracy. Semantic sets were used as an input for the Naïve Bayes classifier. The authors have also developed an automatic tracer tool to resist malware obfuscation. Another experiment presented in (Van Nhuong et al., 2014b) combined three methods including Semantic Set, N-gram byte using the Naïve Bayes Classifier and finally an API function-based signature detection method. The latter being used to perform static detection with the help of a string matching algorithm to reduce the detection time (Navarro, 2001). Combining three different methods helped achieving high accuracy (DR) while the key challenge identified concerns optimisation for real-time detection to reduce time.

E. Other Techniques

Martins et al. (2014) identified nodes on the basis of their relationship and characteristics in a Dependency Graph extracted from an executable file. This would then help to create a model resilient to code mutation. Further, the authors presented a method to extract graphs from binary. The DR achieved was 70%. Another exemplar to this category is a method proposed by Saleh et al. (2011), they have modified a face recognition technique to detect malware. The technique assumes that every face had a linear combination of basic sets, some of these could change over time to some extent, but not all of them. As such, the method has the required capability of measuring the similarity and difference between samples. The authors used a static analysis tool called Principal Composite Analysis and the system database was trained to identify malware. A distance classifier was then computed for the new input file to be checked against the database. If the distance was below the threshold then the input file was considered as malware. The system used a technique called Euclidean Distance to measure the distance of four classes which is good for a small training set but might not deliver the same results with a high number of malware classes (Shanmugam et al., 2013).

IV. CONCLUSIONS AND FURTHER DISCUSSION

The analysis of infection and obfuscation techniques utilised by metamorphic malware is fundamental to develop an effective and efficient detection method. Static Analysis extract information such as device metadata, network connection states and changes to registry files related to suspected software. Whilst Dynamic Analysis, known as Behavioural Analysis, observes the interactions of the malicious file within a Sandbox. A combination of the above two techniques has potential to deliver better outcomes in terms of detection rate and false positives. Two key elements have been extracted, reviewed and examined in this systematic review, namely detection methods and their accompanying statistics (empirical evidence) to evaluate their efficiency. The first technique observed was Opcode based, Operands of Opcode provide additional information about the suspicious file (Zolotukhin and Hamalainen, 2014) which might help in detecting other malware variants. However, this will require large number of labelled executables for each variant, and it is

difficult to acquire such data (Santos et al., 2011). Other challenges include the cost of extra processing time (Santos et al., 2011; Alam et al., 2014b).

Obfuscation is efficient to easily evade detection when Opcode is utilised (Preda, 2012; Mahawer and Nagaraju, 2013) this is mainly due to the change affecting Opcode distribution which thereafter results in the creation of a vague abstract model for metamorphic malware detection (Alam et al., 2014c). Likewise, CFG and API Call Graph based techniques are not effective against encryption/encoding obfuscation techniques (Li et al., 2006; Elhadi et al., 2014). CFG uses static analysis and requires human interaction during the process causing imprecise outcome. Hence, a more dynamic approach is recommended (Moser et al., 2007). Contrary to Opcode, CFG is unable to generate signatures for a file with small foot prints (Paul and Mishra, 2014) which may yield a high FPR. When API Call Graph is used, extracted API call lists and its parameters would be incorrect when the Malware uses encryption. Furthermore, another challenge is related to the extraction process itself using graph construction algorithms (Elhadi et al., 2014) because operating system resources are not included as graph nodes in call graph. API Call Graph depends upon graph construction and may not complete in terms of the number of nodes (Park et al., 2013). Adopting a graph based approach is expensive, complex (Li et al., 2006), and has a problem of NP Complete (Bai et al., 2009; Alam et al, 2014c). Nonetheless, a recent study by Salehi et al. (2017) whose method generated features for both return and argument values of recorded API call lists, obtained DR of 99.9% with less than 1% FPR using 1211/3175 malware and benign samples respectively.

Semantic set is a recently introduced technique to identify metamorphic malware (Van Nhuong et al., 2014a). However, it cannot detect obfuscated malware and requires a lot of human interaction for hex conversion and information manipulation. Therefore, while detection rate is very promising, practical implementation is technically challenging due to the amount of manual input required.

Another recent study by Mehra et al. (2015) combined API Call Graph, CFG and Histogram. The proposed system consisted of three phases. Firstly, the executable is disassembled with a pre-processing algorithm to remove unnecessary statements and to generate CFGs. Next, API Call Graphs are generated with the help of a labelling algorithm. Then, features are generated based on the API Call Graphs. Histograms are created using features same as that of API Call and then used to classify files as either malicious or benign. Khodamoradi et al. (2015) used a decision tree to compute statistics about Opcode and build thresholds. They used a tool called Opcode Statistic Extractor (OSE) to analyse disassembled code and calculate Opcode frequency. This was then fed into a classifier to determine whether the code is malicious. Work on classification or categorization methods is out of scope for this survey, but it is inevitable to include a brief demonstrate on how it can be utilised to counteract malware since it is incorporated within many Hybrid detection approaches as shown below. In principle, the classification process consists of pre-processing and dimensionality reduction techniques, feature selection, feature representation (term weighting), and then selected algorithms (classifiers) are trained on the data set. To increase accuracy, a hybrid approach can also be deployed by integrating multiple algorithms and stemming techniques into this process (Alabbas et al., 2016). O'Kane et al., (2016) proposed using reduced Opcode set for detecting obfuscated malware. In their research, Support Vector Machine was used to classify files. The Opcode data set is created by extracting Opcode density histogram during program execution.

Mirzazadeh et al. (2015) demonstrated how to detect metamorphic malware particularly NGVCK and MWOR using a Linear Discriminant Analysis method. The research framework was based on Opcode Graph Similarity (OGS) of Runwall et al. (2015) which trimmed lifeless ciphers from the graph. Basic Linear Discriminant Analysis (LDA) features such as modifications among class and unpredictability within class were selected as detection criteria. The LDA method used the following phases: pre-processing, training, set threshold and prediction (Raphel & Vinod, 2015). The result of the method obtained detection accuracy rate of 99.7% for MWOR and 100% for NGVCK malwares respectively. Analogous to the LDA based technique is the research of Kuriakose and Vinod (2015) which detected NGVCK and MWORM metamorphic malware with 100% accuracy when 125 features ranking Opcode techniques were deployed to calculate similarity in their file execution. The technique arrived at a Markov Blanket detection correctness of 100% for both NGVCK and MWORM with 1.0 precision. It can be inferred that the LDA technique suppresses feebleness of OGS when results were contributed from all edges and nodes, whereas LDA pruned the junk edges from the graphs thus establishing a distinguished detection threshold between benign programs and metamorphic malware.

Nevertheless, another new proposal by Saleh et al. (2011) utilised a modified version of a face recognition technique to detect malware. This technique uses static analysis which always requires human interaction to train and update data. Any approach that requires direct human interaction is time consuming and should be automated (Moser et al., 2007). This technique is based on pattern matching and therefore liable to issues such as the 2D pattern matching problem in image making.

The included studies emphasised on several key challenges particularly facing the detection process for metamorphic malware, some of which have been briefly discussed in the literature in studies such as (Yoshioka and Matsumoto, 2009) and (Kim and Moon, 2010):

Obfuscation/Evasion. Mainly because obfuscation techniques change the abstract behavioural model used to define the behaviour of a given software. Further, if malware writers have the knowledge of the basic mutation process and detection algorithms used for originating these abstract models, they could amend new designs to evade detection.

Dynamic Analysis. To analyse the behaviour of metamorphic malware, a Sandbox is required to make sure that it would not affect any interconnected environment including the network. The implementation is simpler with static analysis where you can disconnect network connection, while the Internet could be required for dynamic analysis.

Precise Signature. System and Programming languages use control flow and data flow techniques to analyse the behaviour of metamorphic malware. However, these techniques are difficult to apply on metamorphic malware due to its nature of code mutation which will cause time and result in many false positives and false negatives.

Human Interaction. Human input can be fundamental to establish a detection model to work more accurately.

Application Difficulty. When metamorphic malware mutates itself, it becomes very difficult to extract a precise signature that is used in detecting a wide range of malware variants.

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.

Alabbas, W., al-Khateeb, H. M., Mansour, A. (2016), 'Arabic Text Classification Methods: Systematic Literature Review of Primary Studies', 2nd Invited Session on Arabic Natural Language Processing (ANLP): Models, Systems, Data and Applications. October 24-26, 2016 within 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), Tangier, 2016, pp. 361-367. doi:10.1109/CIST.2016.7805072.

Alam, S., Horspool, R.N. and Traore, I. (2013) 'MAIL: Malware Analysis Intermediate Language - A Step towards Automating and Optimizing Malware Detection', 6th International Conference on Security of Information and Networks, pp. 233–240, doi:10.1145/2523514.2527006. doi: 10.1016/j.cose.2014.10.011.

Alam, S., Horspool, N.R. and Traore, I. (2014a) 'MARD: A Framework for Metamorphic Malware Analysis and Real-Time Detection', 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, doi: 10.1109/aina.2014.59.

Alam, S., Sogukpinar, I., Traore, I. and Horspool, R.N. (2014b) 'Sliding window and control flow weight for metamorphic malware detection', Journal of Computer Virology and Hacking Techniques, 11(2), pp. 75–88. doi: 10.1007/s11416-014-0222-y.

Alam, S., Traore, I. and Sogukpinar, I. (2014c) 'Annotated control flow graph for metamorphic Malware detection', The Computer Journal, 58(10), pp. 2608–2621. doi: 10.1093/comjnl/bxu148.

al-Khateeb, H. M., Epiphaniou, G., Alhaboby Z. A., Barnes J., Short E. (2017), 'Cyberstalking: Investigating Formal Intervention and the Role of Corporate Social Responsibility', Telematics and Informatics, Vol. 34, No. 4, pp.339-349. ISSN 0736-5853. doi:10.1016/j.tele.2016.08.016

Bai, L., Pang, J., Zhang, Y., Fu, W. and Zhu, J. (2009) 'Detecting Malicious Behavior Using Critical API-Calling Graph Matching', 2009 First International Conference on Information Science and Engineering, . doi: 10.1109/icise.2009.494.

Carrillo, H. and Lipman, D. (1988) 'The Multiple Sequence Alignment Problem in Biology', SIAM Journal on Applied Mathematics, 48(5), pp. 1073–1082. doi: 10.1137/0148063.

Christodorescu, M. and Jha, S. (2004) 'Testing malware detectors', ACM SIGSOFT Software Engineering Notes, 29(4), p. 34. doi: 10.1145/1013886.1007518.

Dai, S.Y., and Kuo, S.Y. (2007) 'MAPMon: A Host-Based Malware Detection Tool', 17 December 2007. Melbourne, Qld. IEEE. pp. 349–356.

Egele, M., Scholte, T., Kirda, E. and Kruegel, C. (2012) 'A survey on automated dynamic malware-analysis techniques and tools', ACM Computing Surveys, 44(2), pp. 1–42. doi: 10.1145/2089125.2089126.

Elhadi, A.A.E., Maarof, M.A., and Barry, B.I.A. (2013) 'Improving the Detection of Malware Behaviour Using Simplified Data Dependent API Call Graph', International

Journal of Security and Its Applications, 7(5), pp. 29–42. doi: 10.14257/ijsia.2013.7.5.03.

Elhadi, A.A.E., Maarof, M.A., Barry, B.I.A. and Hamza, H. (2014) 'Enhancing the detection of metamorphic malware using call graphs', Computers & Security, Vol. 46, October 2014, pp.62–78, DOI: 10.1016/j.cose.2014.07.004.

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.

Fukushima, Y., Sakai, A., Hori, Y. and Sakurai, K. (2010) 'Smartphone malware detection model based on artificial immune system', Kyoto, 5 October 2010. Institute of Electrical & Electronics Engineers (IEEE). pp. 86–92.

Karim, M.E., Walenstein, A., Lakhotia, A. and Parida, L. (2005) 'Malware phylogeny generation using permutations of code', Journal in Computer Virology, 1(1-2), pp. 13–23. doi: 10.1007/s11416-005-0002-9.

Kim, K. and Moon, B.-R. (2010) 'Malware detection based on dependency graph using hybrid genetic algorithm', Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10, doi: 10.1145/1830483.1830703.

Kitchenham, B. and Charters, S. (2007) Guidelines for performing Systematic Literature Reviews in Software Engineering.

Khodamoradi, P., Fazlali, M., Mardukhi, F. and Nosrati, M. (2015). Heuristic Metamorphic Malware Detection Based on Statistic s of Assembly Instructions using Classification Algorithm. *IEEE* 2015.

Kuriakose, J. and Vinod, P. (2015) 'Metamorphic malware detection: modelling with fewer relevant features and robust feature selection techniques', International Journal of Computer Science, IEEE, Vol. 42, No. 2, pp. 139-151.

Kwon, J. and Lee, H. (2012) 'BinGraph: Discovering mutant malware using hierarchical semantic signatures', 2012 7th International Conference on Malicious and Unwanted Software, doi: 10.1109/malware.2012.6461015.

Lee, J., Jeong, K. and Lee, H. (2010) 'Detecting metamorphic malwares using code graphs', Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, . doi: 10.1145/1774088.1774505.

Li, Z., Sanghi, M., Chen, Y., Kao, M.Y, and Chavez, B. (2006) 'Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience', 2006 IEEE Symposium on Security and Privacy (S&P'06), doi: 10.1109/sp.2006.18.

Lyda, R. and Hamrock, J. (2007) 'Using entropy analysis to find Encrypted and packed Malware', IEEE Security and Privacy Magazine, 5(2), pp. 40–45. doi: 10.1109/msp.2007.48.

Mahawer, D.K. and Nagaraju, A. (2013) 'Metamorphic malware detection using base malware identification approach', Security and Communication Networks, 7(11), pp. 1719–1733. doi: 10.1002/sec.869.

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.

Mehra, V., Jain, V. and Uppal, D. (2015). DaCoMM: Detection and Classification of Metamorphic Malware. pp.668 – 673 IEEE 2015, doi:10.1109/CSNT.2015.62

Mirzazadeh, R., Hossein Moattar, M. and Vafaei Jahan, M. (2015). Metamorphic Malware Detection Using Linear Discriminant Analysis and Graph Similarity, IEEE 2015 fifth International Conference on Computer and Knowledge Engineering, pp.61 – 66, doi:10.1109/ICCKE.2015.7365862.

Moser, A., Kruegel, C. and Kirda, E. (2007) 'Limits of static analysis for Malware detection', Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), doi: 10.1109/acsac.2007.21.

Navarro, G. (2001) 'A guided tour to approximate string matching', ACM Computing Surveys, 33(1), pp. 31–88. doi: 10.1145/375360.375365.

O'kane, P., Sezer, S. and McLaughlin, K. (2016). 'Detecting obfuscated malware using reduced Opcode set and optimised runtime trace. *Springer'*, pp.2 - 10. doi:10.1186/s13388-016-0027-2

Park, Y., Reeves, D.S. and Stamp, M. (2013) 'Deriving common malware behavior through graph clustering', Computers & Security, Vol. 39, November 2013, pp.419–430, DOI: 10.1016/j.cose.2013.09.006.

Paul, S. and Mishra, B.K. (2014) 'Survey of Polymorphic Worm Signatures', International Journal of u and e Service, Science and Technology, 7(3), pp. 129–150. doi: 10.14257/ijunesst.2014.7.3.12.

Preda, M.D. (2012) 'The grand challenge in metamorphic analysis', Information Systems, Technology and Management: 6th International Conference, ICISTM 2012, Grenoble, France, March 28-30, 2012. Proceedings, pp.439–444, DOI: 10.1007/978-3-642-29166-1_42.

Rad, B.B., Masrom, M. and Ibrahim, S. (2012) 'Camouflage in Malware: from Encryption to Metamorphism', International Journal of Computer Science and Network Security, 12(8), pp. 74–83.

Raphel, J. and Vinod, P. (2015) 'Pruned feature space for metamorphic malware detection using Markov Blanket. In *Contemporary Computing (IC3), 2015 Eighth Conference on* (pp. 377-382). IEEE.

Rezaei, F., Hamedi-Hamzehkolaie, M., Rezaei, S. and Payandeh, A. (2014a) 'Metamorphic viruses detection by hidden Markov models', 7'th International Symposium on Telecommunications (IST'2014), doi: 10.1109/istel.2014.7000817.

Rezaei, F., Nezhad, M.K., Rezaei, S. and Payandeh, A. (2014b) 'Detecting encrypted metamorphic viruses by hidden Markov Models', 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), . doi: 10.1109/fskd.2014.6980971.

Runwal, N., Low, R. M., & Stamp, M. (2012). Opcode graph similarity and metamorphic detection. Journal in Computer Virology, Vol. 8, No. 1-2, pp. 37-52.

Saleh, M.E., Mohamed, A.B. and Nabi, A.A. (2011) 'Eigenviruses for metamorphic virus recognition', IET Information Security, 5(4), p. 191. doi: 10.1049/iet-ifs.2010.0136.

Salehi, Z., Sami, A., Ghiasi, M., (2017) 'MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values. International journal of Engineering Applications of Artificial Intelligence', 59(1), pp.95-98.

Santos, I., Brezo, F., Ugarte-Pedrero, X. and Bringas, P.G. (2013) 'Opcode sequences as representation of executables for data-mining-based unknown malware detection', Information,Sciences, Vol. 231, 10 May 2013, pp.64–82, DOI: 10.1016/j.ins.2011.08.020.

Santos, I., Sanz, B., Laorden, C., Brezo, F. and Bringas, P.G. (2011) 'Opcode-sequence-based semi-supervised unknown malware detection', Computational Intelligence in Security for Information Systems: 4th International Conference, CISIS 2011, Held at IWANN 2011, Torremolinos-Malaga, Spain, June 8-10, 2011. Proceedings, pp.50–57, DOI: 10.1007/978-3-642-21323-6_7.

Shanmugam, G., Low, R.M. and Stamp, M. (2013) 'Simple substitution distance and metamorphic detection', Journal of Computer Virology and Hacking Techniques, 9(3), pp. 159–170. doi: 10.1007/s11416-013-0184-5.

Sharma, A. and K. Sahay, S. (2014) 'Evolution and detection of Polymorphic and metamorphic Malwares: A survey', International Journal of Computer Applications, 90(2), pp. 7–11. doi: 10.5120/15544-4098.

Shijo, P.V. and Salim, A. (2015) 'Integrated static and dynamic analysis for Malware detection', Procedia Computer Science, 46, pp. 804–811. doi: 10.1016/j.procs.2015.02.149.

Van Nhuong, N., Nhi, V.T.Y., Cam, N.T., Phu, M.X. and Tan, C.D. (2014a) 'Semantic set analysis for malware detection', Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings, pp.688–700. DOI: 10.1007/978-3-662-45237-0_62

Van Nhuong, N., Nhi, V.T.Y., Cam, N.T., Phu, M.X. and Dang Tan, C. (2014b) 'SSSM-semantic set and string matching based malware detection', the 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), . doi: 10.1109/cisda.2014.7035642.

Vinod, P., Laxmi, V., Gaur, M.S. and Chauhan, G. (2012) 'MOMENTUM: MetamOrphic malware exploration techniques using MSA signatures', 2012 International Conference on Innovations in Information Technology (IIT), doi: 10.1109/innovations.2012.6207739.

Wu, L., Ping, R., Ke, L. and Hai-xin, D. (2011) 'Behavior-based Malware Analysis and Detection', Nanjing, Jiangsu, 24 September 2011. IEEE. pp. 39–42.

Wu, L., Xu, M., Xu, J., Zheng, N. and Zhang, H. (2013) 'A novel malware variants detection method based On function-call graph', IEEE Conference Anthology, doi: 10.1109/anthology.2013.6784887.

Yoshioka, K. and Matsumoto, T. (2009) 'Sandbox Analysis with controlled internet connection for observing temporal changes of malware behavior', The Fourth Joint Workshop on Information Secruity.

Zolotukhin, M. and Hamalainen, T. (2014) 'Detection of Zero-day Malware Based on the Analysis of Opcode Sequences', Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th, 7, pp. 386 – 391. doi: 10.1109/CCNC.2014.6866599.

APPENDIX 1 - Evaluation of selected reviewed papers

This section presents a comparative analysis of studied papers. The purpose of this comparative analysis is to provide a clear view to the reader on strengths and weaknesses of selected reviewed papers. The comparative analysis is shown below in the Table 1.

Table 1. A comparative analysis of studied papers

S#	*Papers*	*Techniques*	*Approach*	*Strengths*	*Limitations*	*Validation*	*Dataset size (# of files)*	*DR%*	*FPR%*
1	(Mahawer and Nagaraju, 2013)	Opcode	The detection method is based on Most Frequently Occurred (MFO) histograms of Opcodes in disassembled files.	Effective against dead code insertion, registry renaming, code reordering.	Cannot detect malware with obfuscation capabilities, this will cause a high rate of FNs.	Validated by experimental result showing significant figures in FPR and DR.	2121	99.5%	0.01%
2	(Alam et al., 2014a)	Opcode	Detecting metamorphic malware by computing weight of MAIL pattern and Control Flow Weight then matching them using index based signature array	Platform independent and support automated analysis with the help of intermediate language.	1- Better detection results are limited to small size dataset 2- Large datasets increase the complexity and is time consuming. 3- Compiler optimisation could affect frequency of Opcode 4- Vulnerable to obfuscation	The results are validated by experiments but 10 fold cross validation technique has been used which makes the results more bias compared to 5 fold cross validation.	1251	99.1%	0.93%
3	(Rezaei et al., 2014a)	Opcode	Detecting metamorphic malware by comparing probability set percentage between sample and virus family using Markov properties.	Proposed method is based on Hidden Markov Model and showed higher efficiency against others antiviruses compared.	Unable to detect unknown metamorphic malware.	Validated by experimental results showing improvement in FPR.	3120	94%	0%

M. Irshad, H. al-Khateeb, A. Mansour, A. Ashawa, and M. Hamisu, "Effective methods to detect metamorphic malware: A systematic review", *International Journal of Electronic Security and Digital Forensics*, vol. 10, no. 2, pp. 138–154, 2018, ISSN: 1751-9128. DOI: 10.1504/IJESDF.2018.090948. [Online]. Available: https://doi.org/10.1504/IJESDF.2018.090948.

S#	*Papers*	*Techniques*	*Approach*	*Strengths*	*Limitations*	*Validation*	*Dataset size (# of files)*	*DR%*	*FPR%*
4	(Vinod et al., 2012)	Opcode	Detecting metamorphic malware using bioinformatics sequence alignment based on Multiple Sequence alignment to align Opcode sequence	Strong logical motivation derived from DNA sequences of inheriting functional, structure similarity from one generation to other same as applied in metamorphic malware	1-Large numbers of mismatch mnemonic pairs cause less Detection Rate. 2-Increased FPR in single signatures, degraded in DR in group and probabilistic signatures.	Validated by experimental results. Shows technique that needs improvement.	724	71%	7%
5	(Rezaei et al., 2014b)	Opcode	Detecting encrypted metamorphic malware by comparing probability set percentage between sample and virus family using Markov properties.	Proposed method is based on Hidden Markov model for encrypted malware and showed higher efficiency against other antiviruses compared.	1- Unable to detect unknown metamorphic malware. 2- Require extensive calculation 3- Vulnerable to obfuscation	Validated by experimental results indicating good efficiency as compare to other antiviruses involved in the experiment.	192	70%	N/A
6	(Alam et al., 2014c)	CFG	Protecting end user from metamorphic malware in real-time	MAIL intermediate language have capability to provide patterns for matching to enhance the metamorphic malware detection. It also capable of platform and analysis independent.	Time consuming while examining large dataset.	Validated by experimental results showing significant/effective DR but FPR is very high.	510	98.9%	4.5%
7	(Eskandari and Hashmi, 2011)	Hybrid	A combination of control flow graph and API Call Graph is used to detect metamorphic malware.	Using semantic aspects, method is capable of detecting obfuscated files.	Slow computational process. Ineffective with non-assembly malware.	Validated by experiments in comparison to different classifier.	4445	97.5%	1.97%
8	(Agrawal et al., 2012)	CFG	To overcome graph comparison problem, normalised metric edit distance technique is employed.	High level semantic signature enable detection of unknown new variants of the same family.	Ineffective with non-assembly malware.	Validated by experimental statistics, showing significant/effective results in FPR but needs more work on Detection.	18	86%	0%

S#	Papers	Techniques	Approach	Strengths	Limitations	Validation	Dataset size (# of files)	DR%	FPR%
9	(Martins et al., 2014)	Dependency Graph	Identifying related nodes on the basis of its relationship and characteristics in dependency graph extracted for an executable file.	It is capable of eliminating the impact of obfuscation techniques on malicious code.	Identifying relationship between nodes and relevant events requires a statistical inference technique.	Validated by experimental results showing an improvement in identification of metamorphic malware compared to the approach used as reference model in (Kim and Moon, 2010).	63	70%	N/A
10	(Elhadi et al., 2014)	API Call	Enhancing API Call Graph construction by integrating API call and system resources with the help of four type of dependencies between nodes.	Sequence profiling & data dependencies to generate accurate call graph. It takes both signature and behaviour input sample.	Dynamic analysis might not explore important API call's execution. Approach has weakness against polymorphic code modification. Time consuming in term of matching process and graph construction.	Validated by experimental results showing improvement in FPR and DR.	514	98%	0%
11	(Kwon and Lee, 2012)	API Call	Transformation of API Call Graphs to sub graphs on the basis of behaviour semantic and related functionalities.	Proposed method has ability to detect many malware variants with the help of few signatures that also reduces signature storage space and analysis time.	Algorithm used for graph extraction cannot build accurate graph from instruction derived from malware sample. Ineffective with non-assembly malware.	Validated by experimental showing significant figures in DR with no FPs.	1863	98%	0%
12	(Wu et al., 2013)	API Call	API Call Graphs are generated and similarity is computed using cosine similarity method.	Proposed method is more robust against obfuscation techniques and effective for malware variants detection.	Ineffective with non-assembly malware. Cosine similarity fails when comparing small graphs.	Validated by experiments performed on prototype system.	More than 200 pair	98%	N/A

S#	*Papers*	*Techniques*	*Approach*	*Strengths*	*Limitations*	*Validation*	*Dataset size (# of files)*	*DR%*	*FPR%*
13	(Lee et al., 2010)	API Call	Mechanism based on semantic characteristics using code graph system before file execution.	Proposed method reduces the number of malware signature by detecting all of the malware variants with single signature of original malware and have ability to defeat evasion techniques.	Weakness in detecting malwares having insertion of useless system calls.	Validated by experiments show low DR at 91% and no availability of FPR.	300	91%	N/A
14	(Van Nhuong et al., 2014a)	Hybrid	Combination of two different methods to build a detection system which inherits the advantages of both methods in detecting malware including obfuscated.	Tracer tool automatically extract semantic sets that overcomes malware obfuscation. Detection is near to perfection.	Processing time needs to be optimised. It does not support real-time detection.	The results are validated by experiments with two different sizes of datasets.	DS1=107 DS2=79	100%	N/A
15	(Van Nhuong et al., 2014b)	Hybrid	Combination of three different methods to build a powerful detection system to detect all type of malware	DR is achieved up to 100%.	It does not support real-time detection. Processing time needs to be optimised.	The results are validated by experiments with two different sizes of datasets.	DS1=107 DS2=79	100%	N/A
16	(Saleh et al., 2011)	Eigenfaces	Based on face recognition technique with some modification assuming that every face has a linear combination of basic set with some changes.	Method is capable of learning new virus patterns for future malware recognition.	1- Time consuming in term of pattern matching process. 2- Distance measuring technique used is simple and limited to high number of dataset for better results.	Validated by experimental results showing 100% in DR but FPR is high.	1250	100%	4%