Hindawi
Security and Communication Networks
Volume 2018, Article ID 1728303, 13 pages
https://doi.org/10.1155/2018/1728303

Research Article

Use of Data Visualisation for Zero-Day Malware Detection

Sitalakshmi Venkatraman [1] **and Mamoun Alazab** [2]

[1]*Department of IT, Melbourne Polytechnic, Prahran Campus, Melbourne, VIC 3181, Australia*
[2]*Charles Darwin University, Darwin, NT 0810, Australia*

Correspondence should be addressed to Sitalakshmi Venkatraman; sitavenkat@melbournepolytechnic.edu.au

Received 24 March 2018; Revised 27 June 2018; Accepted 5 November 2018; Published 2 December 2018

Academic Editor: Qing Yang

With the explosion of Internet of Things (IoT) worldwide, there is an increasing threat from malicious software (malware) attackers that calls for efficient monitoring of vulnerable systems. Large amounts of data collected from computer networks, servers, and mobile devices need to be analysed for malware proliferation. Effective analysis methods are needed to match with the scale and complexity of such a data-intensive environment. In today's Big Data contexts, visualisation techniques can support malware analysts going through the time-consuming process of analysing suspicious activities thoroughly. This paper takes a step further in contributing to the evolving realm of visualisation techniques used in the information security field. The aim of the paper is twofold: (1) to provide a comprehensive overview of the existing visualisation techniques for detecting suspicious behaviour of systems and (2) to design a novel visualisation using similarity matrix method for establishing malware classification accurately. The prime motivation of our proposal is to identify obfuscated malware using visualisation of the extended x86 IA-32 (opcode) similarity patterns, which are hard to detect with the existing approaches. Our approach uses hybrid models wherein static and dynamic malware analysis techniques are combined effectively along with visualisation of similarity matrices in order to detect and classify zero-day malware efficiently. Overall, the high accuracy of classification achieved with our proposed method can be visually observed since different malware families exhibit significantly dissimilar behaviour patterns.

1. Introduction

Malicious software (malware) is a computer program that has the intention of causing harm to the operating system kernel or some security sensitive application or data without the user's consent [1, 2]. Malware includes computer viruses, worms, potentially unwanted programs (PUP), and others that could even compromise a computer. Internet crime using such malware is affecting many businesses and people worldwide. There have been many malicious activities on the web with new attacks caused by unknown variants of existing malware that obfuscate their behaviour to evade from detection [3]. These malware are called zero-day malware (new malware) as there are zero-days between the unknown malware's first attack and the time it is discovered. Such attacks are also called zero-day attacks.

The commonly applied malware detection approaches fall under two main techniques: static and dynamic analysis [4–7]. Static analysis uses the syntax and structural properties of a file by disassembling the program binary in order to extract the features. On the other hand, dynamic analysis of the file is required to be conducted during its running time in order to extract characteristic actions performed by the program. Theoretically, a static analysis is faster and more effective as compared to dynamic analysis due to its advantages from the information captured relating to structural properties such as sequence of byte "signatures" and anomalies in file content. Dynamic analysis can be effective with runtime information, such as running process or by using control flow graph that could be less prone to obfuscated malware. Several previous studies have combined the two approaches for better results [8, 9]. Malware writers use many metamorphic and polymorphic obfuscation techniques such as dead-code insertion, subroutine reordering, code transposition, instruction substitution, code integration, and register reassignment to create variants of an existing malware family in order to evade detection [3, 10]. In addition, packers obfuscate the entire program and ensure that the code can be only

analysed at runtime [11]. Since nonstandard and custom-made packing could make it difficult to disassemble code for reverse engineering, the binary has to be executed in virtual environment for unpacking to perform reverse engineering of the code [12]. Data mining and machine learning techniques have provided promising results to detect such hidden malware effectively in applications over a variety of platforms including smartphones and devices [13–15]. A number of static malware detection approaches have differentiated their work by studying different classifiers such as support vector machine (SVM), k-Nearest Neighbor (KNN), and Naïve Bayes (NB) [16]. Recent studies have also used various data mining techniques for permission usage analysis in mobile applications and the results show that SVM classifier could achieve over 90% accuracy [17]. Overall, several features, from high-level such as API calls or even relevant permissions as well as low-level opcode for n-grams based malware detection, have been explored in previous studies [18–20]. In this work, we have adopted machine learning and similarity mining approaches that have been effectively applied to both static and dynamic malware detection with visualisation as the main focus.

With Big Data and Internet of Things (IoT), the task of a malware analyst becomes highly labour intensive and complex since the existing automatic approaches and techniques are available to detect, identify, or capture only known malware and there is an ever-increasing number of attacks due to unknown obfuscated malware [21]. Even though automated data analysis methods are being developed to mimic this process as much as possible, they still require the domain experts to correct and disambiguate intermediate results [22]. Malware analysts or domain experts are required to analyse large volumes of executable codes, transaction logs, or network traffic data to identify anomalies as existing automated analysis techniques cannot replace them in detecting zero-day malware. The use of visualisation could be considered to support this analysis process of detecting suspicious unknown malware that exhibits anomalous behaviour patterns. Visualisation techniques could be used to effectively intertwine human and computerized analysis processes to provide malware analysts with a powerful visual tool using visual representations of data as an effective user interface. The key advantage of visualisation is its capability of presenting huge amounts of data in a more intuitive and interactive manner.

Some recent studies have delved on visualisation techniques to speed up the malware detection process significantly [23, 24]. Visual analytics suit Big Data contexts where complex data requires data analysis to combine automation with analytical reasoning of human experts. In visual analytics, similarity mining is a machine learning method based on the analysis of similarities of the distance measures and has been recently adopted to detect malware. In this paper, we provide a visualisation of the similarity matrix between different malware programs that are commonly employed by attackers. In addition, we use the visualisation technique to compare malware dataset with benign dataset to demonstrate their significant difference in behaviour patterns.

In summary, the main contribution of this paper is due to a signature-free anomaly based detection method using visualisation techniques. The key objective of the proposal is to cope with packed and polymorphic transformations and metamorphic obfuscations of malware for achieving effective and efficient solutions in order to address the zero-day malware detection problem. The approach uses the knowledge of normal behaviour patterns of the Application Programming Interface (API) and investigates patterns of obfuscated code further by adopting several data mining techniques of extracted features and statistical n-gram opcode analysis with innovative techniques to classify whether the binary content is benign or malicious. The knowledge of API function calls features along with various distance measures of vector models are used in a visual way to detect obfuscated malware families.

We have organised the overall structure of the paper as follows. In Section 2 we discuss the commonly used visualisation techniques in the computer security field. In Section 3, we describe how the proposed method adopted for this study combines both frequency and knowledge of binary features as well as the use of key similarity measures for visualisation and thereby differentiate this study uniquely from previous research investigations. Section 4 presents the experimental results of investigating our visualisation approach with datasets containing both malware and benign files. An in-depth analysis of how our approach achieves high classification accuracy using machine learning algorithms is presented along with limitations and challenges of the study. Finally, we provide our conclusions and future research work in Section 5.

2. Visualisation Techniques for Computer Security

In general, visualisation using similarity techniques falls under two main categories: (1) projection-oriented or (2) semantic-oriented [25]. Text visualisation techniques are predominantly used in many applications with five main purposes: (1) to visualise document similarity, (2) to identify content, (3) to reveal sentiments and emotions in text, (4) to explore document corpus, and (5) to analyse various domain-specific rich-text corpus, such as social media data, online news, emails, poetry, and prose. Various application domains could employ visual analytics to reap the benefits of visualisation, including malware analysis.

In the field of computer security, visualisation tools have evolved over a period of time and they are becoming more useful with Big Data and processing large files. State-of-the-art techniques for malware analysis can be found in literature [2], and these fall under two main categories of static and dynamic analysis [3–6]. Using such techniques, malware analysts can analyse a file in a hex editor with relative ease, but such tools do not provide the structure of the log contents or a relationship between the data in the logs. Two-dimensional visualisation of a similarity matrix is a traditional technique used in both static and dynamic analysis to capture the relevant similarity measures between

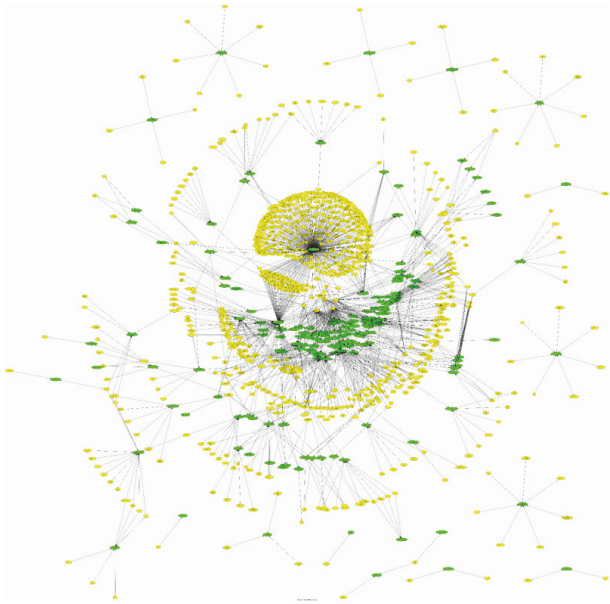


Figure 1: Visualisation of SSH brute force attempts.

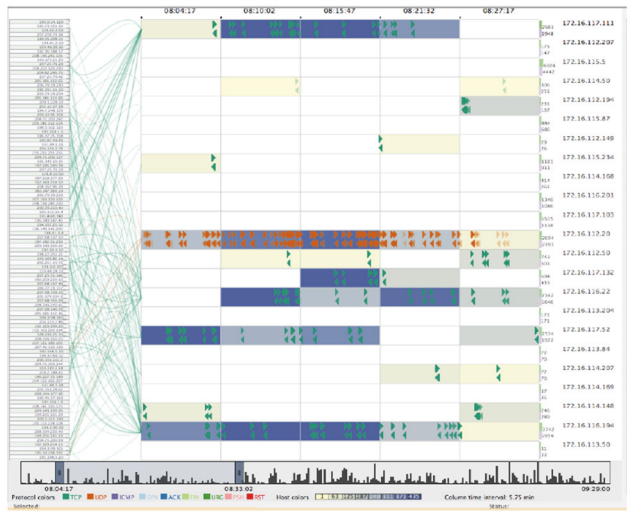


Figure 2: Visualisation matrix of a network host showing port activities and packets with zoom levels.

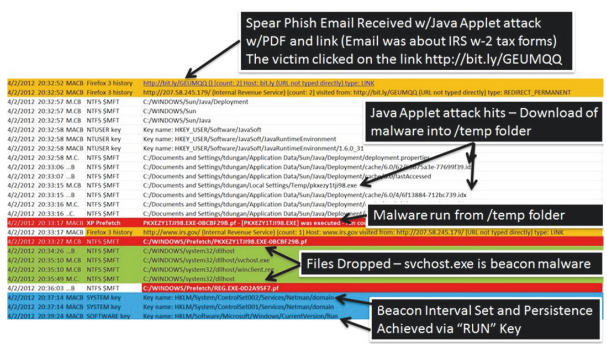


Figure 3: Visualisation of spear phishing attack timeline.

objects. It provides three key properties: (i) once the similarity space is formed, the high-dimensionality of the data does not affect further processing; (ii) clusters of equal importance get formed; and (iii) clusters that are related to one another are shown adjacent to each other aiding in visualisation of results [25–27]. The advantage of visualisation is to make a judgment for cluster enhancements. A commonly found application is at the document level, where similarities of content are visually represented for summarizing document collections. It is a common practice to visually represent documents as points on either a 2D or 3D plane. The distance between each pair of points shows how similar the two documents are; i.e., the closer they are, the more similar the two document contents are [28].

Recently, some research studies have employed visualisation in the analysis of network security attacks [29]. For example, Figure 1 shows a visualisation of secure shell (SSH) brute force attempts and one could zoom in to different colour-coded areas for investigating the details of UserIDs and Internet Protocol (IP) addresses related to various anomalies [30].

Visualisation techniques can be used to display an overview of large packets at a time as well as to show the relationships between network packets and allow analysts to zoom into interesting sections to see more detailed data. Figure 2 shows such features with a visualisation matrix of a network host displaying port activities and a table of packets [31].

Next, it is also possible to use visualisation to conduct timeline analysis to explain the chronology of a malware attack [32]. For example, Figure 3 shows a timeline analysis flowchart of different stages in a spear phishing attack with colours indicating which stages were successful.

After identifying such anomalies and attacks, more critical information could be extracted using colour-coded visualisation of the different characteristics and types of connections to the attacked system. Figure 4 shows the information on "what," "where," and "when" of these connections and how the distances to other hosts could be estimated using their IP addresses [33, 34]. Different types of alerts are shown as separate sectors of concentric rings in consecutive time intervals and possible attacks depicting many connections to the same host.

Next, malware analysts are required to analyse the malware codes that caused the attack and classify them for a proactive prevention of future attacks. Malware coders modify small parts of the source code of an existing malware to produce a new variant or obfuscated malware resulting in zero-day attacks. For instance, the register reassignment transformation replaces code between registers by exchanging register names with no other effect on program behaviour [3]. If register ebx is dead throughout a given live range of the register eax, it can replace eax in that live range. The signature that encodes [PUSH ebx] is not the same as the one that encodes [PUSH eax] and hence becomes obfuscated. Register reassignment such as replacing [PUSH ebx] with [PUSH eax]

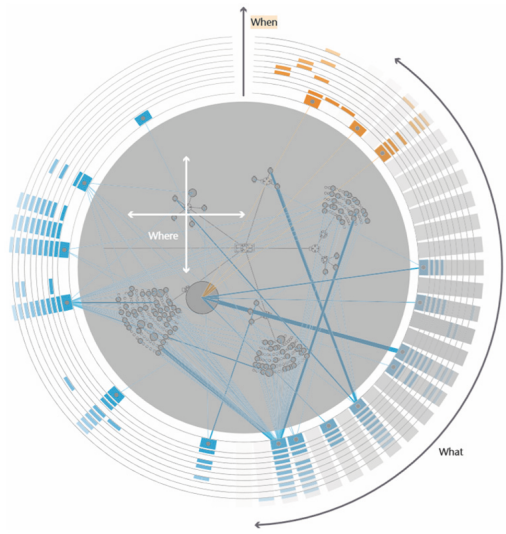


Figure 4: Visualisation of network connection types (what), resources (where) attacked, and the time (when).

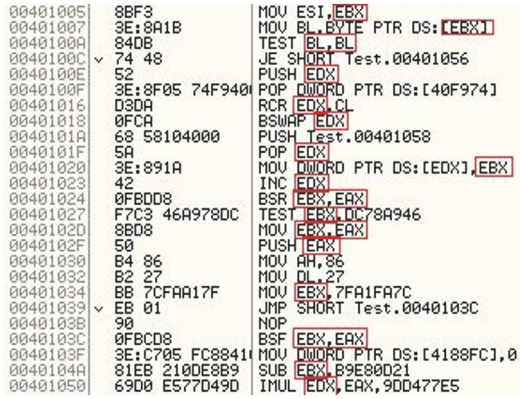


Figure 5: Text visualisation of register reassignment.

to exchange register names and text visualisation to identify such an obfuscated malware is shown in Figure 5.

While text visualisation is time-consuming in today's world of Big Data, texture-based analysis of image visualisation of different sections of the executable binary codes could improve the productivity of a malware analyst. The advantage of images used in visualisation is that they can give more information about the structure of the malware and could display even small code changes while retaining the whole structure of the code. For example, Figure 6 shows different sections of the code to have unique texture that help in identifying similar patterns [35]. Malware variants belonging to the same existing malware family exhibit similar images as observed in Figure 7 for a known malware Dialplatform.B (a) and Fakerean (b).

There are limitations with texture-based image analysis of malware. Such an approach cannot be applied to analyse behaviour patterns for detecting obfuscated code changes. In addition, malware could be packed using different packing methods and with different resolution, which leads

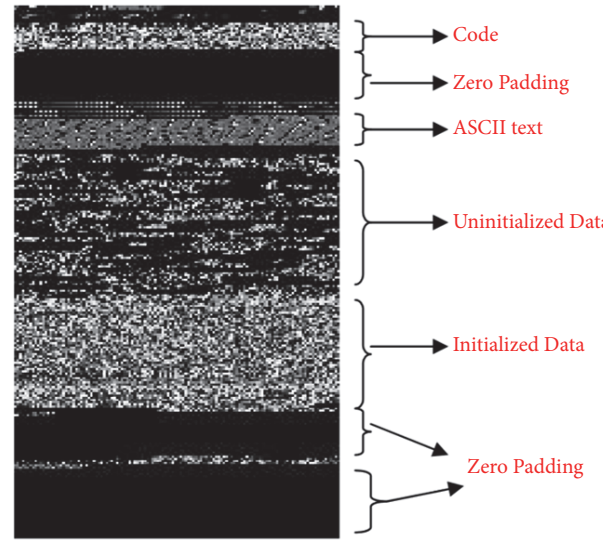


Figure 6: Visual representation of the sections of malware code.

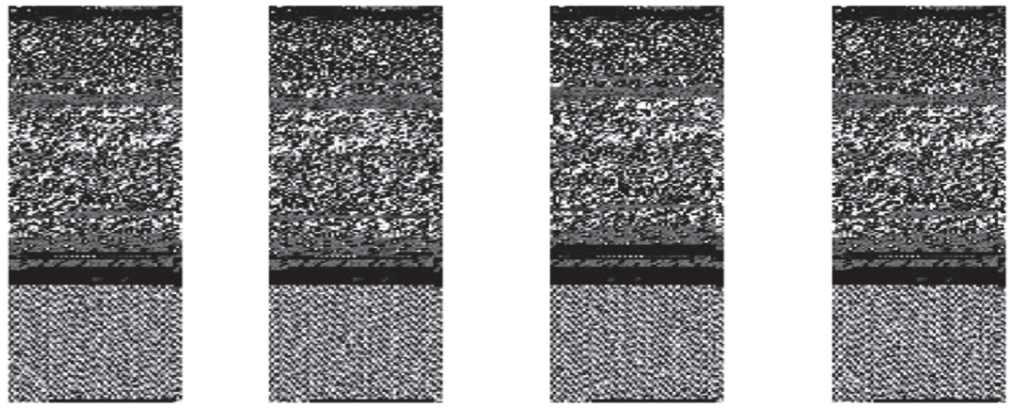
to ambiguity in the classification of malware unknown. These factors make texture based visualisation technique not always reliable. This warrants more research work towards a robust method of classifying malware using visualisation techniques [36]. Much research studies reported in the literature are quite restricted with a focus on network traffic and infiltration analysis [37–39]. In this paper, we demonstrate how visualisation of similarity mining could be applied for detection as well as classification of zero-day malware. We innovatively apply visualisation of similarity matrices for an accurate detection and classification of unknown malware. Our proposed technique is described in the next section.

3. Proposed Method Using Similarity Mining

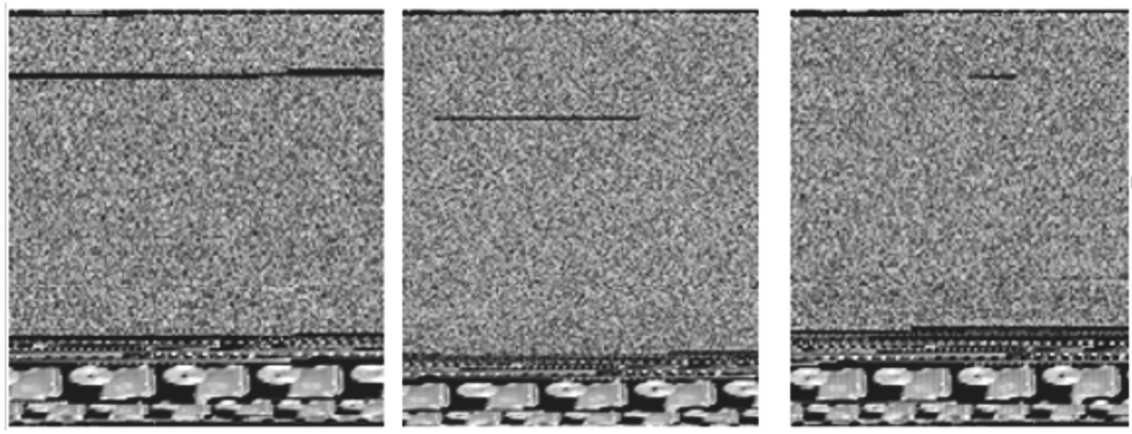
The accuracy of malware detection techniques is based on how well the behaviour patterns exhibited by malicious code could be extracted and correlated. In general, the intrusion techniques and attack methods adopted by malware could be broadly classified as static, dynamic, and hybrid [40, 41]. While static approaches use code syntax manipulations, dynamic approaches use process changes. In some cases, hybrid methods combine both code manipulation and process changes. For easy and quick implementation, malware code writers adopt one of the main forms of automatic generation of new malware resulting in zero-day attacks as listed below:

(i) Installation or software bundling (static): malicious code is plugged into host software or bundled in additional components by taking advantage of an installation exploit. The malicious code runs every time the software/component is used and it gets installed into a system and modifies that system.

(ii) Self-replication (static): malware gets onto new targets that are associated with an existing one.



(a) 4 variants of Win32.Dialplatform.B



(b) 3 variants of Fakerean

Figure 7: Visual similarity in different variants of a known malware.

(iii) Scanning or surveying (dynamic): malware could be operating from remote, finding new targets for its attack

(iv) Injection into process or data (hybrid): the maliciousness gets injected into other running processes or data so as to gain additional privileges.

(v) Concealment (hybrid): this method is used to hide the presence of certain processes, files, or system resources or to prevent the disabling of software, processes, or security settings.

(vi) Payload (hybrid): this method is used to download or to send content (personal data, processes, and behaviour patterns) from or to third parties.

Investigating a compromise requires the understanding of the above forms of malware implementations. Rootkits, log files, password/hidden files, and processes are some of the avenues through which compromises could be analysed contextually using automatic and semiautomatic tools primarily based on the patterns exhibited by malware [42]. However, with the present Big Data scenario, such analysis becomes complex and time-consuming to undergo a comprehensive and thorough investigation of malware samples and to accurately detect and classify zero-day malware [43, 44]. We propose a visualisation of malware behaviour patterns to address this problem.

Our proposed method is based on the premise that visualisation can be used to support both individual behaviour analysis of a malware sample and accurate malware classification of a new malware (zero-day malware). The malware classification uses comparison of malware samples to identify common behaviour exhibited by known malware families. As exemplified in previous section, there are two broad categories of visualisation techniques adopted for malware comparison, namely, image-based and feature-based [10, 45]. Image-based techniques make use of visual images of either binary data or behaviour logs of the malware samples [46]. Images generated in this approach are similar to those shown in Figures 1 and 2, where visual mappings are used to generate an image for each malware sample. Feature-based technique compares different malware samples based on extracted features [10]. Though this approach could be harder to compare a large set of characteristics with each other quickly using a visual overview, various visual analytics techniques and tools are available to let the user filter, search, compare, and explore a wide range of patterns occurring during malware comparisons [47, 48].

Previous studies have considered a combination of both image-based and feature-based technique for malware classification without execution or disassembly of malware code as shown in Figure 7 [35]. However, due to their inability to analyse behaviour patterns of malware and their limitation on operating with only selected file formats and packing methods, we propose a new visualisation technique using similarity matrices of features depicting behaviour patterns of malware as well as displaying them in image form for faster analysis. In another related work, opcode sequences are converted into RGB pixels in an image matrix and the similarity of image matrices is computed [49]. Our approach is different in two ways based on enhancements with previous work [18, 19]. Firstly, we make use of a huge dataset of about 52,000 malware samples for our study while the previous work experiments with only 290 malware samples with 16 families. Secondly, using similarity matrices, we adopt a hybrid approach with feature-based technique for direct comparison of various features, which helps to understand which features correlate any two malware binaries and which do not [20, 50]. At the same time, we project these similarity matrices as image patterns for a faster identification and classification of new malware into their correct malware families.

We model our proposed malware analysis method to consist of three main stages as follows:

Stage 1: preprocessing stage

Stage 2: feature processing stage

Stage 3: visualisation stage

By following through these three stages, we derive four modelling steps that are described next.

Stage 1 (preprocessing stage). Stage 1 involves preprocessing of the dataset to identify packed (compressed) files since malware writers adopt packing of binaries for employing polymorphic obfuscation of malicious code in order to evade detection [11, 12]. Most recent malware programs are generated with various packing techniques making it very difficult to detect using static or dynamic analysis. We have adopted multiple techniques of packed binary detectors to separate packed and unpacked files from the dataset [19, 20]. In the experiment conducted on the dataset of about 52,000 samples, the result at the preprocessing stage indicates that about 77% of malware programs were packed and 23% were unpacked.

We converted the malware files into images, extracted features of these images from a pretrained deep convolutional neural networks (CNN) model, and then embedded these into 2 dimensions using t-Distributed Stochastic Neighbor Embedding (t-SNE), so we can visualize. We then clustered the malware in image feature space, using k-means; Figure 8 shows what clustering algorithm outcome looks like for 6 different families. Different colours correspond to different clusters by the k-means.

Stage 2 (feature processing stage). In Stage 2, all files are unpacked in order to disassemble the binary executable to retrieve the assembly program. This stage involves deobfuscating and reverse-engineering the program codes and applying feature extraction techniques effectively to conduct feature analysis using data mining techniques. The details of the feature processing stage in Stage 2 consist of first two modelling steps as shown below.

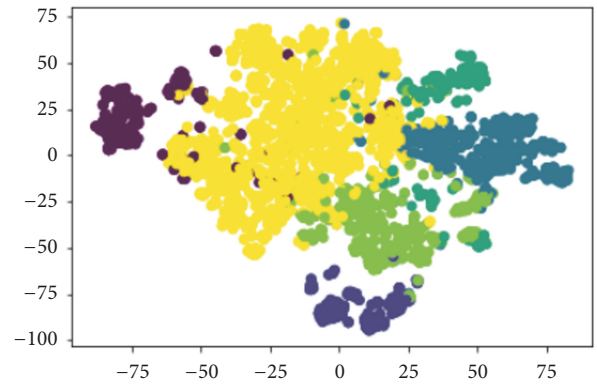


Figure 8: Simulated data with six clusters.

Step 1. The executable program is disassembled using reverse engineering tools. Each disassembled executable (P) represents a vector of functions x, y. (P') is the variant malware of the original executable (P). Each function is represented as an array of vector of functions. All executable programs, malicious or benign, have the goal to perform an action using API function calls. In this step, we extracted API calls and important machine-code features from the assembly program. A statistical analysis of the Windows API calling sequence reflects the behaviour of a particular piece of code. We identified commonly used API function call features that are based on the malicious behaviour patterns to predominantly fall under six groups, namely, "search files," "çopy/delete files," "get file information," "move files," "read/write files," and "change file attributes." We adopt intelligent extraction of the behaviour of features of API function calls that relate to (i) hooking of system services, (ii) creating or modifying files, and (iii) getting information from the file for changing information about the DLLs loaded by the malware. Binary n-gram features are also extracted for analysis. We perform n-gram statistical modeling to obtain the distribution of the executables for n-values ranging from 1 to 5. Extracting binary n-gram features to complement the API call features has uniquely helped to train the classifier correctly. We adopt a supervised learning approach that uses a dataset to train, validate, and test an array of classifiers, which results in building a model using support vector machine (SVM). The model is measured based on factors such as accuracy, false positives, and false negatives. In addition, the model needs to be tested against larger sets of malware samples for verifying the accuracy of the modelled system. Overall, this step is used for achieving a robust identification of malicious code as against benign code using SVM to train the classifier as part of the machine learning process. The extracted features undergo a statistical test to determine the malware class accurately based on suspicious behaviour patterns.

Step 2. The similarity between the functions of programs (P) and (P') is computed using similarity mining of different distance measures. A similarity matrix is generated for each comparison. After obtaining the API sequence from binaries, the signature database is updated based on these API calls. This sequence is compared to a sequence or signature (from the signature database) and is passed through the similarity measure module to generate the similarity report. Different distance measures are implemented and similarity analysis is performed by using eight commonly used distance measures in vector models, namely, Cosine, Bray-Curtis, Canberra, Chebyshev, Manhattan, Correlation, Euclidean, and Hamming distance similarity measure for Nearest Neighbor (NN). The definitions of these measures are provided below.

In similarity mining, we implemented eight different similarity metrics in vector models and similarity analysis was performed with distance measures such as Cosine, Bray-Curtis, Canberra, Chebyshev, Manhattan, Correlation, Euclidean, and Hamming distance. We provide below the details of these eight metrics adopted in this study.

The distance measures defined under each of the eight metrics use the following variables:

(i) u is the opcode sequence of the test file

(ii) v is the opcode sequence of a file in the database

(iii) n is the vectors dimension

(iv) D is the distance between vectors u and v

3.1. Cosine Distance. The Cosine distance computed between two n-vectors u and v is defined as

$$D = 1 - \frac{uv^T}{||u||_2 \, ||v||_2} \tag{1}$$

3.2. Bray-Curtis Distance. The Bray-Curtis distance measured between two n-vectors u and v is defined as

$$D = \frac{\sum |u_i - v_i|}{\sum |u_i + v_i|} \tag{2}$$

3.3. Canberra Distance. The Canberra distance between two n-vectors u and v is defined as

$$D = \frac{\sum_i |u_i - v_i|}{\sum_i |u_i + v_i|} \tag{3}$$

3.4. Chebyshev Distance. The Chebyshev distance computed between two n-vectors u and v is defined as

$$D = \max_i |u_i - v_i| \tag{4}$$

3.5. Manhattan Distance. The Manhattan distance between two n-vectors u and v is defined as

$$D = \sum_i |u_i - v_i| \tag{5}$$

3.6. Correlation Distance. The correlation distance computed between two n-vectors u and v is defined as

$$D = 1 - frac\,(-\overline{u})\,(v - \overline{v})^T \, \|(u - \overline{u})\|_2 \, \|v - \overline{v}\|_2^T \tag{6}$$

where $\overline{u}$ is the mean of a vectors elements and n is the common dimensionality of u and v.

3.7. Euclidean Distance. The Euclidean distance between two n-vectors u and v is defined as

$$D = \|u - v\|_2 \tag{7}$$

3.8. Hamming Distance. The Hamming distance between two n-vectors u and v is simply the proportion of disagreeing components in u and v and is defined as

$$D = \frac{C_{01} + C_{10}}{n} \tag{8}$$

where C_{ij} is the number of occurrences of $u[k] = i$ and $v[k] = j$ for $k < n$

Stage 3 (visualisation stage). Stage 3, namely, the visualisation stage, consists of the last two modelling steps of our proposed model as shown below.

Step 3. The similarity matrix values are then compared with the threshold values. Different colour schemes depict different distances from threshold values. The image patterns are used to determine if the executable is malicious or not. Comparison of the values with other samples can help to identify groups or malware families. The classification methods require training data to validate the models formulated in arriving at the threshold values for the similarity matrix. Therefore, K-fold cross-validation has been used for evaluating the results of a statistical analysis generating an independent dataset using 10 folds. Having k=10 folds using 90% of full data is used for training (and 10% for testing) in each fold test. Evaluation (feature selection + classification) was done inside 10-fold cross-validation loop on all malware and benign dataset. Then SVM is applied to the training data with the goal to produce a model, which is then used to predict the target of the test data. In order to achieve a higher accuracy of the predictive model for generalisation, K-fold cross-validation approach is used and applied for test data, with k=10. This value is commonly used to estimate how well the trained SVM model is going to perform in the future.

Step 4. Benchmarking of the results is conducted in this step. Different similarity mining metrics are adopted, and their performances are compared. A minimum of eight distance measures was used as similarity metrics in this step. The classification algorithm follows a supervised learning algorithm with four different variations to validate the results obtained. Among the four basic types of kernels used by SVM to map the training vectors to the N-dimensional space, the Radial Basic Function (RBF) kernel is applied, as it can handle the nonlinear cases. Classification performance is

tested based on $1/\sigma^2$ and C parameters from (9) given below, where $C > 0$ is the penalty parameter of error term.

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma^2}\right) \quad (9)$$

The accuracies achieved for malware classifications are compared based on the following standard measures:

(1) True positive (TP): number of correctly identified malicious codes

(2) False positive (FP): number of wrongly identified benign codes, when a detector detects benign file as a malware

(3) True negative (TN): number of correctly identified benign codes

(4) False negative (FN): number of wrongly identified malicious codes, when a detector fails to detect malware

The efficiency of the proposed method is evaluated using the following performance measures:

Positive (P): the predicted attribute belongs to the right class.

$$P = TP + FN \quad (10)$$

Negative (N): the predicted attribute belongs to the wrong class.

$$N = FP + TN \quad (11)$$

Overall accuracy: percentage of correctly identified code, given by

$$\begin{aligned} Overall\ Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ &= \frac{TP + TN}{P + N} \end{aligned} \quad (12)$$

In summary, malware writers make use of metamorphic and polymorphic engines to generate new dissimilar malware variants for zero-day attacks. A "similarity analysis" can quantify the level of similarity and the difference between two binary executables. Our proposed malware detection method is based on the degree of similarity of the extracted Win API function calls and opcode sequence features between malware and benign files. The maliciousness of a code is estimated using the eight distance measures given in (1)–(8) with support vector machine (SVM) mining algorithms and the classification performance is measured using (9). Similarity based detection is well-suited for static metamorphic and polymorphic malware analysis since new malware programs are generated as variants of existing ones to achieve zero-day attacks. In previous research studies, API calls have been analysed as well as how they could be used to profile malware [18–20]. In this study, we enhance the recent research work [50] in terms of addition of visualisation features. Further, we have conducted validation of results resulting in high accuracies for malware classification and these findings are presented in the next section.

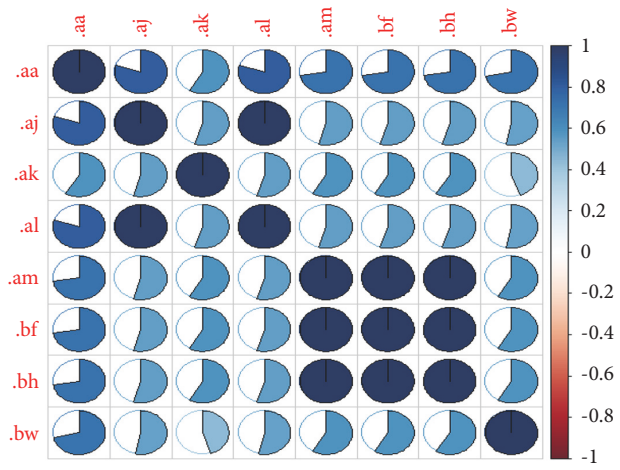


Figure 9: Similarity matrix of the malware familyTrojan.Downloader.Win32.Dadobra.

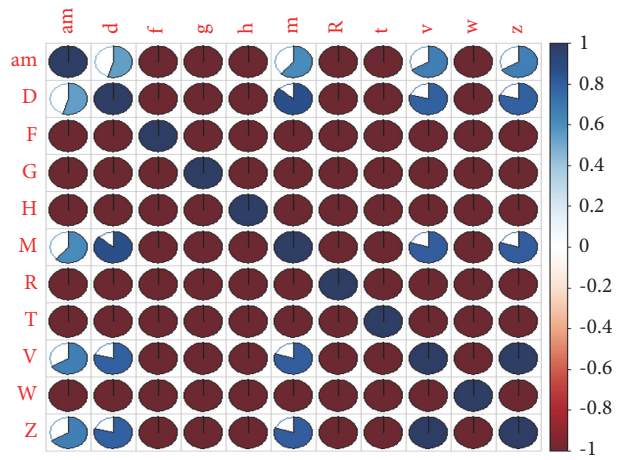


Figure 10: Similarity matrix of the malware family Worm.Win32.Delf.

4. Experimental Results and Discussion

The experimental investigation of the similarity analysis was carried out by implementing distance measures and analysis of the various data mining algorithms in Python Programming Language. The experiment was run in three different processors, which aided in the effective malware classification, and was evaluated using very large real-life malware dataset consisting of about 75,000 samples obtained through public databases such as VX heavens [51]. More than two-thirds of the samples were malware and the remaining were benign samples. The similarity distance system developed in this research was able to automatically identify all malware variants. Figures 9 and 10 provide an illustration of the similarity matrices for malware in the same family. In these matrices, the similarity measures calculated are colour-coded based on the distance from the threshold values. These distance measures can take values between 0 and 1, blue colour or positive correlations are displayed in blue which means high similarity, and red colour is low or zero similarity.

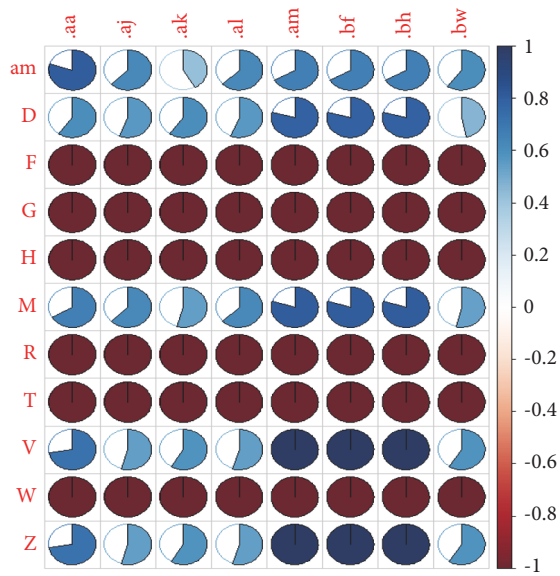


Figure 11: Similarity matrix of two malware families Trojan.Downloader.Win32.Dadobra vs Worm.Win32.Delf.

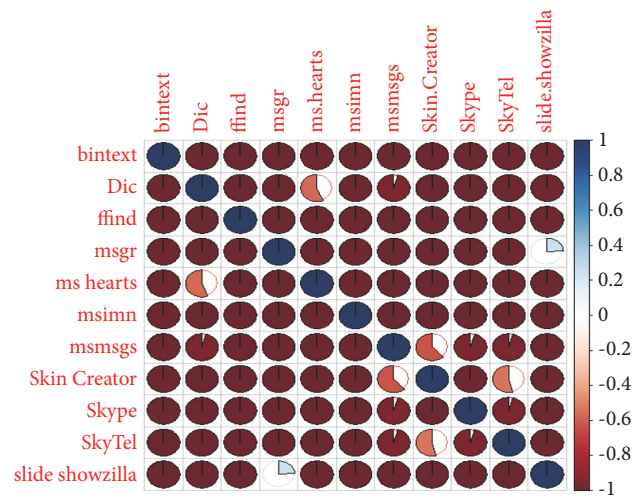


Figure 12: Similarity matrix of benign files.

Color intensity and the size of the circle are proportional to the correlation coefficients. In order to scale the values from $[0, 1]$ to $[-1, 1]$ and to reverse direction since 0 was similar in original data, $((1 - M) - 0.5) * 2$ was used. We used the similarity generated data to visualise the correlation matrix using Correlogram. R corrplot function is used to plot the graph of the correlation matrix.

From the visualisation analysis, the security analyst can see the entire cell in Figure 9 to have close similarity of the malware Win32.Dadobra (a Trojan), and Figure 10 can be detected as a variant of the original malware Win32.Delf (a Worm). Further, we conducted experimental comparisons between any two malware families to understand whether their behaviour patterns are similar. As an example, Figure 11 shows the similarity matrix between two different malware families, Win32.Dadobra and Win32.Delf, and Figure 12 shows the similarity matrix obtained for all the benign files.

The similarity matrix for two malware datasets from different families can be easily visualized in Figure 11. The visualisation results from Figures 9, 10, and 11 show that there is a low distance/high similarity between malware variants but not with the benign programs. Figure 12 demonstrates that there is high distance/low similarity between the benign datasets.

Table 1 provides the mean values obtained for each of the eight distance measures applied for the entire dataset. Analysing the overall results of the similarity metrics in Table 1, we observe that the Euclidean similarity metric strongly differentiates with mean values falling far apart while performing similarity comparisons between (i) malware-benign, (ii) malware-malware, and (iii) benign-benign. It would be an interesting investigation to explore further on the comparative performance of these metrics, which is beyond the scope of this study. Overall, our experiments demonstrate that the proposed model finds high similarities between malware variants but not with the benign programs, which makes it easy to differentiate even unknown malware from benign ones and classify them accurately.

Using hybrid visualisation approaches of feature-based and image-based analysis shown in Figures 9. 10. 11, and 12 along with Table 1, it can be seen that similarity mining is very efficient and effective to detect malware variants from the same family or different families of malware. Also, the experiments confirm that there is no similarity among the different benign files, but they exhibit a similar image representation of similarity matrix, which is uniquely different from that of malware. This important observation is as follows: it is very hard to find any image similarity between the malware dataset and the benign dataset which validates that the proposed system is able to clearly distinguish between malware and benign datasets.

In the classification algorithms, the training data and testing data were selected by making a partition on the database of malware and benign files for carrying out the experiments. We adopted the most common type of cross-validation, namely, k-fold cross-validation that is a standard practice adopted in similar research studies adopted for many classifiers [52, 53]. For the similarity mining, we adopted Sequential Minimal Optimization (SMO) algorithm in support vector machine (SVM) method with 4 different kernels; (i) SMO-Normalized Polynomial Kernel Function, (ii) SMO-Polynomial Kernel Function, (iii) SMO-Radial Basis Function (RBF), and (iv) SMO-Pearson VII Kernel function (PUK). The advantage of SMO is its ability to solve the Lagrange multipliers analytically with fast implementation of support vector machines. Further, it is a popular supervised learning algorithm used for classification and regression problems. In Figure 13, the overall accuracy rate for malware detection achieved using the four kernels of SMO for our experimental datasets is shown. Normalized Polynomial Kernel provides the highest accuracy for all the k cross-validations, with $k = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. In particular, with k=10, we achieved about 98.6% accuracy for malware

Table 1: Mean measures obtained using the eight similarity metrics.

Distance Method	Malware – Benign	Malware – Malware	Benign – Benign
Cosine	0.34	0.29	0.39
Bray Curtis	0.84	0.77	0.86
Canberra	0.84	0.77	0.86
Chebyshev	61.27	31.45	79.98
Manhattan	14.32	96.24	18.03
Correlation	0.35	0.243	0.403
Euclidean	78.94	44.31	106.39
Hamming	0.034	0.04	0.03

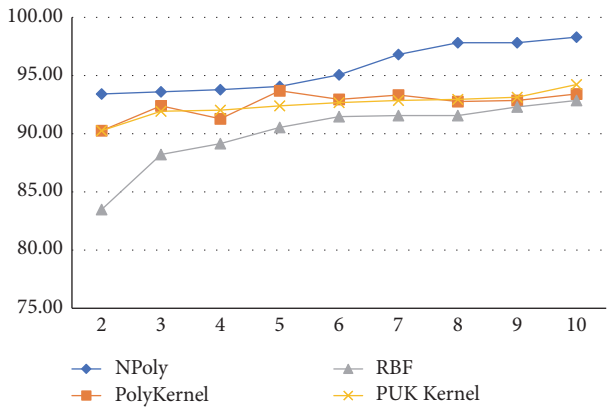


Figure 13: Accuracy of malware classification using SMO with k cross-validations (k=2 to 10).

detection which is among the best so far reported in literature using large datasets.

In visualisation of malware comparisons, one of the major challenges faced is to deal with unanticipated patterns that may appear and that would require further investigation and analysis [54–56]. In addition, with the proliferation of IoT devices, the application layer is prone to malware attacks due to their increasing popularity and platform accessibility. Typically, an IoT device application layer includes local web applications, cloud-based applications, and smartphone apps that are accessible to numerous third party app markets leading to security threats [57, 58]. Hence, multiple IoT malware attacks are possible and these fall under two main categories according to the way in which IoT malware infects devices: (i) by brute force attacks through a dictionary of weak usernames and passwords; (ii) by exploiting unfixed or zero-day vulnerabilities found in IoT devices [43]. With Big Data and IoT, the malware datasets could be complex and unstructured that require more dynamic and scalable visualisation and more efficient feature extraction [44]. We anticipate further enhancing our visualisation framework to address these challenges in the future. Next, we provide the limitations of the current study and key challenges that would trigger future research directions.

4.1. Limitations and Challenges. Today, we witness an explosion of Internet of Things (IoT) worldwide with millions of security devices connected via the Internet every day. Hence, there is a rapidly increasing threat from malware attackers warranting an efficient monitoring of vulnerable systems. These heterogenous devices are collecting mountains of data collected from computer networks, servers, and mobile devices leading to Big Data environment. Efficient monitoring and analysis of such Big Data for malware proliferation are gaining importance. In such IoT environment of Big Data evolving in the recent years, blockchain technology is being adopted to protect the integrity of data storage and ensure process transparency [59]. However, open challenges still exist in this direction, and this paper does not delve into the intersection of our proposed zero-day malware detection and visualisation approach with blockchain technology.

Another limitation of this study is in the coverage of juice filming charging (JFC) attacks that can steal users' sensitive and private information from smartphone devices during phone charging in public places such as airports and shopping malls. Since such attacks take place during the charging period when the users' information can be leaked through a standard micro USB connector without the need for any permission or installation of apps, the increase in the processor usage could be studied to identify the attack [60]. However, it is not within the scope of this paper and visualisation techniques for detecting the suspicious behaviour of smartphones during charging would pose another challenging problem.

From recent literature we find that there is a need to provide visual representations appropriate for IT-security experts with the ability to externalize knowledge for sharing purposes. Some developments towards a knowledge-assisted visualization system for behavior-based malware analysis have been reported [61]. While this research work has not considered knowledge externalization methods, with rising Big Data infrastructure, future malware analysis process would depend heavily on knowledge-based visual analytics techniques.

5. Conclusions and Future Work

This paper proposed a new hybrid method of feature-based and image-based visualisation of similarity mining to identify and classify malware accurately. Our visualisation technique is effectively used to compare malware samples for better communication of their behaviour patterns and

faster detection and classification of new malware (zero-day malware). We calculated the similarities between the malware variants using eight different distance measures to generate similarity matrices and to identify the malware family by adopting visualisation of the distance scores. The experimental study of our proposed method involved large datasets of about 75,000 samples with more than two-thirds consisting of malware samples and benign samples forming the rest. By performing similarity mining of the innumerable obfuscations of extended x86 IA-32 (opcodes) found in these malware samples, we were successfully able to detect and classify unknown malware that had escaped from traditional detection methods. The proposed method is efficient and accurate in identifying malware visually due to three main properties observed through our experimental results:

(1) Malware opcodes exhibit significant dissimilarity of behaviour patterns as compared to the benign opcodes and hence result in very high true positives

(2) For malware programs belonging to the same family, the uniqueness and closeness in similarity can be visually deciphered through the colour-coded distance measures of the similarity matrix and each malware family exhibits a unique visual pattern of the similarity matrix. This property warranties correctness in assigning any new malware to its original malware family from where it was obfuscated

(3) The image of the similarity matrix for benign codes is unique with distance measures either close to 0 (red) or close to 1 (yellow) as shown in Figure 12. This property helps in accurately identifying benign codes thereby resulting in almost zero false positives.

We have also performed a comparison of the most commonly used classifier, namely, Sequential Minimum Optimisation (SMO) algorithm of support vector machine (SVM) with four different kernels such as Normalised Polynomial Kernel, Polynomial Kernel, Radial Basis Function (RBF), and Pearson VII kernel function (PUK). The data mining based detection system implemented for this study to detect obfuscated malware has achieved high true positive (TP) rate of about 98.6% and low false positive (FP) rate of less than 2%, which has not been achieved in literature so far. With almost 99% accuracy achieved in the case of SMO-Normalised Polynomial Kernel, we envisage that our visualisation approach using similarity mining would effectively differentiate the behaviour patterns of zero-day malware and would enable security analysts to detect and classify new malware (zero-day malware) quickly and accurately. These results are much higher than those reported in literature; this paper has taken a step further in contributing to the evolving realm of visualisation techniques used in the information security field.

As future work, the results of similarity mining forming images of matrices would be analysed in arriving at image or graphical-signatures for different malware families. Tools could further be developed to process these image patterns as graphical-signatures to detect malware variants automatically. In addition, the proposed similarity mining based detection and classification of malware could be studied in terms of efficiency in the security analysts' speed and performance within real-time environments. Variations in the parameters such as distance measures, SVM kernels, and image colours used could be studied for their effect, and these investigations could lead to the proposal and use of more advanced feature selection techniques. Research findings in this direction would play a key role when more computing memory and time for processing the extracted features are required by very large datasets due to Big Data and IoT predictions of the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This research did not receive specific funding but was performed as part of the employment of the authors.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] J. Aycock, "Computer Viruses and Malware," in *Advances in Information Security*, Springer-Verlag, New York, NY, USA, 1st edition, 2006.

[2] G. Mohamed and N. B. Ithnin, "Survey on Representation Techniques for Malware Detection," *System American Journal of Applied Sciences*, 2017.

[3] I. You and K. Yim, "Malware obfuscation techniques: a brief survey," in *Proceedings of the 5th International Conference on Broadband Wireless Computing, Communication and Applications (BWCCA '10)*, pp. 297–300, November 2010.

[4] M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," in *Proceedings of the 12th conference on USENIX Security Symposium USENIX Association*, vol. 12, Washington, DC, USA, 2003.

[5] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys*, vol. 44, no. 2, article 6, 2012.

[6] M. Akour, I. Alsmadi, and M. Alazab, "The malware detection challenge of accuracy," in *Proceedings of the 2nd International Conference on Open Source Software Computing, OSSCOM '16*, Lebanon, December 2016.

[7] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 29–46, 2017.

[8] M. Alazab, S. Huda, J. Abawajy et al., "A Hybrid Wrapper-Filter Approach for Malware Detection," *Journal of Networks*, vol. 9, no. 11, 1969.

[9] M. Alazab and S. Venkatraman, "Detecting malicious behaviour using supervised learning algorithms of the function calls," *International Journal of Electronic Security and Digital Forensics*, vol. 5, no. 2, pp. 90–109, 2013.

[10] A. Malhotra and K. Bajaj, "A hybrid pattern based text mining approach for malware detection using DBScan," *CSI Transactions on ICT*, vol. 4, no. 2-4, pp. 141–149, 2016.

[11] T. M. Brosch, *Maik Runtime packers: The hidden problem. Black Hat*, and M.. Maik Runtime packers: The hidden problem. Black Hat, USA, 2006.

[12] L. Martignoni, M. Christodorescu, and S. Jha, "OmniUnpack: Fast, generic, and safe unpacking of malware," in *Proceedings of the 23rd Annual Computer Security Applications Conference, ACSAC 2007*, pp. 431–440, USA, December 2007.

[13] A. Pektaş, M. Çavdar, and T. Acarman, "Android Malware Classification by Applying Online Machine Learning," in *Computer and Information Sciences*, vol. 659 of *Communications in Computer and Information Science*, pp. 72–80, Springer International Publishing, Cham, 2016.

[14] M. Chowdhury, A. Rahman, and R. Islam, "Malware Analysis and Detection Using Data Mining and Machine Learning Classification," in *International Conference on Applications and Techniques in Cyber Security and Intelligence*, vol. 580 of *Advances in Intelligent Systems and Computing*, pp. 266–274, Springer International Publishing, Cham, 2018.

[15] A. Bhattacharya and R. T. Goswami, "DMDAM: Data Mining Based Detection of Android Malware," in *Proceedings of the the first international conference on intelligent computing and communication*, J. K. Mandal, S. C. Satapathy, M. K. Sanyal, and V. Bhateja, Eds., vol. 458, pp. 187–194, springer, Singapore, 2017.

[16] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, pp. 1–22, 2018.

[17] L. Sun, Z. Li, Q. Yan, W. Srisa-an, and Y. Pan, "SigPID: significant permission identification for android malware detection," in *Proceedings of the 2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 1–8, Fajardo, PR, USA, October 2016.

[18] M. Alazab, M. Al Kadiri, S. Venkatraman, and A. Al-Nemrat, "Malicious code detection using penalized splines on OPcode frequency," in *Proceedings of the 3rd Cybercrime and Trustworthy Computing Workshop, CTC '12*, pp. 38–47, Australia, October 2012.

[19] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures," in *Proceedings of AusDM2011 Ninth Australasian Data Mining Conference*, Ballarat, 2011.

[20] M. Alazab, "Profiling and classifying the behavior of malicious codes," *The Journal of Systems and Software*, vol. 100, no. 2, pp. 91–102, 2015.

[21] S. Venkatraman, "Autonomic Framework for IT Security Governance," *International Journal of Managing Information Technology*, vol. 9, no. 3, pp. 1–11, 2017.

[22] M. Wagner, W. Aigner, A. Rind et al., "Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis," in *Proceedings of the the Eleventh Workshop on Visualisation for Cyber Security*, pp. 9–16, ACM, Paris, France, November 2014.

[23] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1496–1519, 2014.

[24] N. Cao, L. Lu, Y.-R. Lin, F. Wang, and Z. Wen, "SocialHelix: visual analysis of sentiment divergence in social media," *Journal of Visualization*, vol. 18, no. 2, pp. 221–235, 2015.

[25] N. Cao and W. Cui, *Introduction to Text Visualization*, Atlantis Press, Paris, 2016.

[26] D. Keim, "Information visualisation and visual data mining," *IEEE Transactionson Visualisation and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[27] S. Few, *Information Dashboard Design - The Effective Visual Communication of Data*, Sebastopol, CA: O'Reilly, 2006.

[28] N. Diakopoulos, D. Elgesem, A. Salway, A. Zhang, and K. Hofland, "Compare clouds: visualizing text corpora to compare media frames," in *Proceedings of IUI Workshop on Visual Text Analytics*, 2015.

[29] H. Shiravi, A. Shiravi, and A. A. Ghorbani, "A survey of visualization systems for network security," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, 2012.

[30] W. B. Balakrishnan, *Security Data Visualisation*, SANS Institute Inc, 2014.

[31] T. Y. Zhang, X. M. WangLi, Z. Z. Li, F. Guo, Y. Ma, and W. Chen, "survey of network anomaly visualisation," *Scince China Information Sciences*, vol. 60, no. 12, 2017.

[32] W. Shanks, *Enhancing Intrusion Analysis through Data Visualisation*, SANS Institute, Inc, 2015.

[33] S. Foresti, J. Agutter, Y. Livnat, S. Moon, and R. Erbacher, "Visual correlation of network alerts," *IEEE Computer Graphics and Applications*, vol. 26, no. 2, pp. 48–59, 2006.

[34] M. Wagner, D. Sacha, A. Rind et al., "Visual Analytics: Foundations and Experiences in Malware Analysis," in *book: Empirical Research for Software Security: Foundations and Experience*, L. ben Othmane, M. Gilje Jaatun, and E. Weippl, Eds., pp. 139–171, CRC/Taylor and Francis, 2017.

[35] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security, (VizSec '11)*, USA, July 2011.

[36] T. Songqing, *Imbalanced Malware Images Classification: a CNN based Approach. CoRR abs/1708.08042*, 2017.

[37] G. Conti, *Security data visualisation - graphical techniques for network analysis*, No Starch Press, San Francisco, 2007.

[38] R. Marty, *Applied security visualisation. Upper Saddle*, Addison-Wesley, River, NJ, 2009.

[39] J. Jacobs and B. Rudis, "Data-driven security analysis, visualisation, and dashboards," in *Indianapolis*, John Wiley & Sons, 2014.

[40] R. Erbacher, K. Walker, and D. Frincke, "Intrusion and misuse detection in large-scale systems," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 38–47.

[41] S. Venkatraman, "Autonomic context-dependent architecture for malware detection," in *Proceedings of. International Conference on e-Technology*, Singapore, 2009.

[42] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi, "Focusing on context in network traffic analysis," *IEEE Computer Graphics and Applications*, vol. 26, no. 2, pp. 72–80, 2006.

[43] A. Wang, R. Liang, X. Liu, Y. Zhang, K. Chen, and J. Li, "An Inside Look at IoT Malware," in *Industrial IoT Technologies and Applications*, vol. 202 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 176–186, Springer International Publishing, Cham, 2017.

[44] R. Gove, J. Saxe, S. Gold, A. Long, and G. Bergamo, "SEEM:A scalable visualisation for comparing multiple large sets of attributes for malware analysis," in *Proceedings of the 11th*

Workshop on Visualisation for Cyber Security, VizSec, ACM, 2014.

[45] S. Z. M. Shaid and M. A. Maarof, "Malware behavior image for malware variant identification," in *Proceedings of the 4th International Symposium on Biometrics and Security Technologies, ISBAST '14*, pp. 238–243, Malaysia, August 2014.

[46] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, pp. 1–14, 2014.

[47] A. Long, J. Saxe, and R. Gove, "Detecting malware samples with similar image sets," in *Proceedings of the the Eleventh Workshop*, pp. 88–95, Paris, France, November 2014.

[48] K. Han, J. H. Lim, and E. G. Im, "Malware analysis method using visualization of binary files," in *Proceedings of the the 2013 Research in Adaptive and Convergent Systems*, pp. 317–321, Montreal, Quebec, Canada, October 2013.

[49] K. Han, B. Kang, and E. G. Im, "Malware Analysis Using Visualized Image Matrices," *The Scientific World Journal*, vol. 2014, Article ID 132713, 15 pages, 2014.

[50] S. Venkatraman and M. Alazab, "Classification of Malware Using Visualisation of Similarity Matrices," in *Proceedings of the 2017 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 3–8, London, November 2017.

[51] VX Heavens. (n.d.). https://archive.org/download/vxheaven-windows-virus-collection.

[52] X. Wang, D. Zhang, X. Su, and W. Li, "Mlifdect: Android Malware Detection Based on Parallel Machine Learning and Information Fusion," *Security and Communication Networks*, vol. 2017, Article ID 6451260, 14 pages, 2017.

[53] J. Yan, Y. Qi, and Q. Rao, "Detecting Malware with an Ensemble Method Based on Deep Neural Network," *Security and Communication Networks*, vol. 2018, Article ID 7247095, 16 pages, 2018.

[54] S. Gad, W. Javed, S. Ghani et al., "ThemeDelta: Dynamic segmentations over temporal topic models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 5, pp. 672–685, 2015.

[55] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Malicious behavior patterns," in *Proceedings of the 8th IEEE International Symposium on Service Oriented System Engineering, SOSE '14*, pp. 384–389, UK, April 2014.

[56] T. K. Dang and T. T. Dang, "A survey on security visualization techniques for web information systems," *International Journal of Web Information Systems*, vol. 9, no. 1, pp. 6–31, 2013.

[57] B. Kang, S. Y. Yerima, K. McLaughlin, and S. Sezer, "N-opcode analysis for android malware classification and categorization," in *Proceedings of the 2016 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2016*, UK, June 2016.

[58] C. Bae and S. Shin, "A collaborative approach on host and network level android malware detection," *Security and Communication Networks*, vol. 9, no. 18, pp. 5639–5650, 2016.

[59] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," *IEEE Access*, vol. 6, pp. 10179–10188, 2018.

[60] W. Meng, L. Jiang, Y. Wang, J. Li, J. Zhang, and Y. Xiang, "Jfcguard: Detecting juice filming charging attack via processor usage analysis on smartphones," *Computers & Security*, 2017.

[61] M. Wagner, A. Rind, N. Thür, and W. Aigner, "A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS," *Computers & Security*, vol. 67, pp. 1–15, 2017.