Journal of Cyber Security
DOI: 10.32604/jcs.2022.033537

Tech Science Press

Article

A Survey on Visualization-Based Malware Detection

Ahmad Moawad*, Ahmed Ismail Ebada and Aya M. Al-Zoghby

Computer Science Department, Faculty of Computers and Artificial Intelligence, Damietta, New Damietta, 34517, Egypt
*Corresponding Author: Ahmad Moawad. Email: ahmad.moawad@outlook.com

Received: 01 September 2022; Accepted: 02 October 2022

Abstract: In computer security, the number of malware threats is increasing and causing damage to systems for individuals or organizations, necessitating a new detection technique capable of detecting a new variant of malware more efficiently than traditional anti-malware methods. Traditional anti-malware software cannot detect new malware variants, and conventional techniques such as static analysis, dynamic analysis, and hybrid analysis are time-consuming and rely on domain experts. Visualization-based malware detection has recently gained popularity due to its accuracy, independence from domain experts, and faster detection time. Visualization-based malware detection uses the image representation of the malware binary and applies image processing techniques to the image. This paper aims to provide readers with a comprehensive understanding of malware detection and focuses on visualization-based malware detection.

Keywords: Malware detection; malware image; malware classification; visualization-based detection; survey

1 Introduction

Malware is a serious global issue, and the rapid growth of malware attacks has made it one of the primary risks to computer systems. Norton labs [1] reported 9 million threats on average every day, and McAfee labs [2] reported 688 attacks per minute and 40 threats per minute in the first quarter of 2021. Malware (Malicious Software) is any malicious code that compromises data security and impairs the system's confidentiality, integrity, functionality, and reputation, typically without the user's knowledge. Malware has several broad categories, including viruses, worms, trojans, and ransomware; each category is organized into families based on the type of variants.

Malware has disastrous effects, demanding a new technique for detection that is effective against multiple malware variants and is also quick. Traditional anti-malware software is incapable of detecting malware variants and instead relies on a signature database to identify malware by looking up malicious patterns. Conventional techniques such as static analysis, dynamic analysis, or hybrid analysis (static and dynamic) are time-consuming and require domain expertise. Recently, visualization-based malware detection has become popular because of its accuracy, detect known and unknown malware variants, domain expert independence, and faster detection. It uses the image representation of the malware binary, unlike conventional detection methods.

This research covers the following: malware types, malware analysis techniques, malware detection techniques, and visualization-based detection. This paper compares and contrasts different strategies, with a particular emphasis on visualization-based malware detection.

2 Malware Types

Malware may have various objectives, including stealing, damaging, disrupting, locking data, allowing remote access, and bypassing the computer system's access constraints. Malware types continue to grow and adapt; they share certain characteristics and vary in others. The following list summarizes the most prevalent malware types:

2.1 Virus

It is host-dependent, self-replicating, and spreads by injecting code from one program to another and from one computer to another. Like human viruses, they require a device to live on and are activated through user action, such as double-clicking on the executable file [3].

2.2 Worms

It is self-contained and host-independent and is classified into two broad categories: network worms which exploit the network's vulnerabilities to spread and infect other networks, and email worms which spread and infect others via email. It replicates itself and is activated without user action.

2.3 Trojan

Trojan or Trojan horse is disguised as legitimate software and hides its real purpose. Once installed by the user without any knowledge, it allows remote access to the computer system and executes actions designed to steal, destroy, or disrupt data. Unlike viruses, it does not self-replicate [3,4].

2.4 Ransomware

Ransomware or crypto-malware is malware that encrypts data and does not allow the victim to use the operating system until the victim pays the attacker a ransom to regain access. Attackers have recently begun using bitcoin to receive money rather than typical bank transactions to hide their identity [5].

2.5 Botnet

Bots, spiders, web bots, or crawls are internet robots. Once installed on the victim's device, the bots can collect data and passwords. A botnet malware controls many computers and the infected systems; attackers can use these powerful computers to launch attacks against specific servers, such as Distributed Denial of Service (DDoS) [6].

2.6 Adware

It involves advertising and continuously displays unwanted advertisements. It collects data about a user's online browsing history to target advertisements. The most common motivation for Adware is to collect data about the user to earn advertising revenue [5,7–15].

2.7 Spyware

It keeps track of a user's login credentials, such as usernames and passwords, or collects sensitive data. Keyloggers are common spyware that records keystrokes on the keyboard and captures the login credentials [7–10].

2.8 Rootkits

Rootkits are a sort of malware designed to provide the hacker with remote access and control of the victim's device without the victim's awareness. This type of malware is designed to remain hidden and operate for an extended period of time, causing various types of damage. Once an attacker gains access to computers, he/she can steal data or use the victim's device to carry out a criminal act, such as using a computer as part of a botnet to launch a Distributed Denial of Service (DDoS) attack [6,16].

2.9 Fileless Malware

The name indicates that this malware does not require a download or installation but exploits operating system applications. For example, when a victim hits a specific link, the memory is loaded, allowing the attacker to load codes that remotely capture the victim's data.

2.10 Malversting

Similar to Adware, the difference is that it originated from reputable websites such as email services that allow for advertisement spaces. The space could be used to insert harmful images or code. The user places complete trust in the legitimate website and may click on advertisements that infect the computer the same way spyware does [17].

In addition to the previously mentioned malware types, there is a blended attack or combined attacks [11] that combines or mixes various malware types, such as viruses, worms, and trojans, in order to exploit one or more vulnerabilities in the targeted systems. Cisco report [12] shows that crypto-mining, phishing, trojans, and ransomware were the top active threats in 2020. Crypto-mining [13] is another malware attack that exploits the victim's resources, such as processors, graphics cards, and network bandwidth, for illicit cryptocurrency mining. Due to the popularity of cryptocurrency, the attacker attempts illicit mining to profit from the blockchain. Cryptocurrency mining demands a large amount of power and resources to compete with other computers worldwide by trying to solve some challenges from the blockchain [14]. The first computer able to solve the challenge will be rewarded, i.e., 6.25 bitcoin (BTC) in 2021 [18]. According to the Cisco report, crypto-mining is one of the most common attacks in the manufacturing industry [12].

3 Malware Analysis

Malware analysis is essential for developing effective malware detection techniques and tools. It is the process of analyzing how the malware runs, behaves, and defines its characteristics. The outcome is to build anti-malware tools to protect against malware attacks or remove malware from infected machines. The analysis of the malware can be non-visual such as static, dynamic, hybrid, and memory. Different types of analysis exist, all of which attempt to explain how the malware operates. Fig. 1 illustrates the different malware analysis approaches.

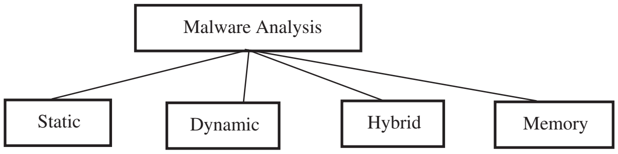


Figure 1: Malware analysis techniques

3.1 Static Analysis

It is a technique for analyzing the malware file without running it. Static analysis exists in two types: non-visual and visual malware. Non-visual static analysis is performed by translating the executable binary to assembly code, studying the reversed assembly code, and looking for patterns. The binary is converted into assembly code using a disassembler, and the code is analyzed using a debugger and code analyzer. Analyze the patterns in headers, signatures, string sequence n-grams, API calls, control flow graph (GFG), operation code (opcode) frequency, and byte sequence [4,19].

The advantages of non-visual static analysis are that it is fast and safe, and the primary issue with static analysis is obfuscation, which malware authors employ to make analysis more difficult. Additionally, the task is time demanding and requires domain expertise to perform the analysis. The static analysis uses reverse engineering tools like x64dbg, IDA Pro, Ollydbg, Ghidra, Cutter, Binary Ninja, Gnu debugger, and Radare2 [8,9].

Static visual analysis is used to transform the binary executable file into an image and study that image to capture similar patterns to malware by using graph entropy, image metrics, and image processing techniques. Different techniques are used to capture the malware patterns from binary, like Scale Invariant Feature Transform (SIFT), Dense SIFT (DSIFT), Scale Invariant Feature Transform (SURF), GIST, Principal Component Analysis (PCA), Discrete Wavelet Transform DWT, deep Convolutional Neural Networks (CNN). Static visual analysis is fast, less time-consuming, and does not require any domain experts. It is able to detect unknown and obfuscated malware [20].

3.2 Dynamic Analysis

Dynamic analysis or behavioral analysis works by executing the malware executable inside an isolated and controlled environment like a simulator, emulator, virtualization, etc., and tracing the behavior of the malware, function calls, function parameters, instruction traces, and control flows using tools like Process Explorer, Process Monitor, etc. [8]. The advantages of this analysis are that it can detect known and unknown malware as well as obfuscated malware. The disadvantages of this analysis are that it is hard and takes a lot of time to extract dynamic features, and malware can discover that it is running inside a virtual environment and change its behavior or hide its actual conduct.

3.3 Hybrid Analysis

This technique combines static and dynamic approaches to overcome their limitations and improve detection accuracy as well as the ability to detect unknown and obfuscated malware. First, examine the malware's signature, then run it inside a controlled environment and observe its behavior.

3.4 Memory Analysis

It analyzes the infected machine's memory by taking a memory dump. Memory analysis has two steps: memory acquisition is getting a memory image dump from the infected machine's memory using tools such as Memdump, LiME, Volatilitux, Memdump, etc., and memory analysis is examining the memory image for malicious activity using tools like Volatility, Rekall, etc. [8,9] by looking at the process list and associated threads, system calls, kernel hooks, API hooks, and network interfaces. Malware authors' use techniques to defeat memory analysis like memory hiding and memory acquisition failure to overcome memory forensics. Recently, memory analysis has gained popularity for detecting malware due to its efficiency and accuracy [4].

4 Malware Detection Techniques

Malware detection techniques are necessary to safeguard systems and machines from malware infections by determining whether or not a piece of software or a file includes malware [21]. Different malware detection exit like signature-based, heuristic-based, specification-based, and visualization-based Fig. 2.

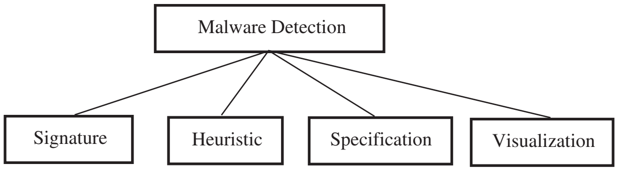


Figure 2: Malware detection techniques

4.1 Signature-Based Detection

The majority of antivirus software detects malware using a signature-based approach. This technique extracts a signature from a malware file and uses it to identify similar malware. The antivirus program scans and generates signatures for each file and compares them to already known malware's signatures in its database. It constantly monitors and scans files and network traffic for signature matches. If any match a signature in its database, the antivirus will flag the file or network traffic as a threat and block it from performing actions. The antivirus vendors update signatures periodically when they discover new malware. This technique is effective and fast in detecting known malware. The weakness is that attackers could change the malware signature and evade the malware detection; changing the malware's signature makes the antivirus tool unable to detect unknown malware that does not have a signature in the database [7,21].

4.2 Heuristic-Based Detection

Heuristic-based, anomaly-based, or behavior-based detection examines the behavior of known and unknown malware. System's behavior is observed, and a record of important information is kept that may be validated and checked in the case of an attack. The reason for keeping the system's behavior in a record is to compare this behavior when unexpected. Three main components in heuristic-based approaches:

- Data collection which collects and gathers the data, which may be static or dynamic.
- Interpretation which interprets and converts the collected data in the data collection step.
- Matching algorithm which matches the behavior's signature with the converted data in the interpretation.

This technique can detect both known and unknown malware. The downsides are that it must constantly update the system profile's data, which may be enormous, resulting in using resources heavily, such as CPU, memory, and disk space, plus a high false-positive rate [16].

4.3 Specification-Based Detection

The specification-based detection technique examines applications with their specifications, looking for expected and unexpected behavior. It involves monitoring program executions and detecting deviations in their behavior from the specification rather than detecting the occurrence of a specific attack pattern. This detection method is derived from heuristic-based detection. The distinction between heuristic-based and specification-based detection techniques is that heuristic-based detection techniques use Machine Learning (ML) and Artificial Intelligence (AI) to determine a legitimate program and indicate its normal and unexpected behavior. On the other hand, specification-based detection techniques examine the behavior specified in the system specification. It lowers the false-positive rate while increasing the false-negative rate [3].

4.4 Visualization-Based Detection

It has recently gained popularity due to its ability to detect malware accurately, reduce detection time, independence from domain experts and feature engineers, and detect a large number of malware files, as will be discussed in detail in Section 6.

5 Malware Authors' Techniques

Malware authors are aware of the detection process by using anti-malware software or conventional analysis. And to make anti-malware detection difficult and make the reversed code as unmeaningful as possible, the authors employ obfuscation techniques. Originally obfuscation intended to safeguard developers' intellectual property, malware authors use these techniques to evade detection [22].

Obfuscation techniques can be divided into two categories: anti-static and anti-dynamic analysis techniques [23]. Anti-static methods include instruction reordering, dead-code insertion, code packing, register reassignment, subroutine reordering, and instruction substitution. Anti-dynamic methods include: detecting the controlled virtual environment and debuggers. These techniques enable the author to create multiple malware variants by altering the signature and bypassing anti-malware detection. They allow the malware authors to convert a program to a new distinct version while retaining its functionality as usual.

6 Visualization-Based Detection

Instead of focusing on detection based on the non-visible features in conventional malware detection techniques, we convert the malware binary into an image and study that image by extracting features, then feed extracted features into a classifier to check if it is benign or malware. Once the malware binary is converted to the image, we are dealing with an image processing problem, and we can employ various techniques for extracting features and classifications. This detection approach combines visualization technology with a different method such as machine learning, deep learning, or both.

Yoo [24] (2004) represented and visualized the embedded viruses in the executable files using Self-Organizing Maps (SOM). "The idea of the research is based on the fact that executable codes and embedded viruses have different structures", and the SOM algorithm can find these differences.

Conti et al. [25] (2008) proposed a method for converting malware binary files into a grayscale image representation and claimed that visual analysis of malware images assists in differentiating data regions in the malware images. This experiment was the first to employ visualization by converting each Byte into a pixel of range 0 and 255. They claimed that binary files are a heterogeneous set of raw data such as images, text, etc., which means when applying changes techniques like compression, encoding, and encryption, the visual of each file can represent a unique pattern [26].

Nataraj et al. [27] (2011) suggested converting the malware binaries into grayscale images, extracting texture features from images using the GIST algorithm, and passing the feature vector to the K-Nearest Neighbor (KNN) to classify the malware. The results looked good in terms of accuracy and taking less time, which encouraged many researchers to work in this area.

The noticed challenges in the visualization approaches are collecting appropriate malware dataset samples or using a small dataset without causing overfitting to the model. Visualization-based detection could use machine learning, deep learning, or both with image processing and computer vision techniques. The general architecture of any visualization-based malware detection Fig. 3 consists of main steps:

1. Convert malware binary into images
2. Extract features from converted malware images
3. Train the model and classify with extracted features

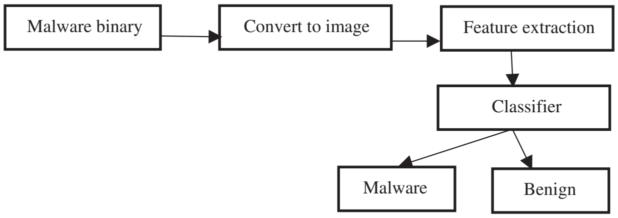


Figure 3: The general architecture of visualization-based detection

These steps are shared between studies and the differences in converting malware binary into images, the representation of the image grayscale or color, image format, feature extraction, and classifier steps.

6.1 Converting Binary to Image

The first step in visualization-based detection is converting an executable binary file into an image representation. The output images are meaningless to humans, but a computational method can contrast their features. Several ways exist for converting binary to an image, including the one proposed by Nataraj et al. [27]; the binary bit string is divided into substrings, and each substring is the 8-bit length which is a Byte. Then convert each Byte into a decimal value between 0 and 255. For example, the following bit string 0110110101001111 → 01101101, 01001111 = 109, 97. A binary number with eight-length $B = (I7,\ I6,\ I5,\ I4,\ I3,\ I2,\ I1,\ I0)$ can be converted into a decimal value D using:

$$D = I_7 * 2^7 + I_6 * 2^6 + I_5 * 2^5 + I_4 * 2^4 + I_3 * 2^3 + I_2 * 2^2 + I_1 * 2^1 + I_0 * 2^0 \qquad (1)$$

After the conversion, the malware binary file is converted to a one-dimension vector of decimals. The one-dimension vector is converted to a two-dimension matrix of the specified width. Then the grayscale image is formed from the matrix. In Table 1 the empirical observation of the recommended image widths for various file sizes. Fig. 4 Illustrates the steps for converting malware to grayscale images.

Table 1: Image width for different images sizes

File size (KB)	Image width
<10	32
10–30	64
30–60	128
60–100	256
100–200	384
200–500	512
200–1000	768
>1000	1024

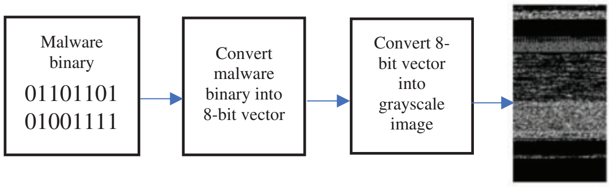

Malware binary
01101101
01001111
Convert malware binary into 8-bit vector
Convert 8-bit vector into grayscale image

Figure 4: Malware binary conversion to a grayscale image

In addition to the mentioned method, there are two other methods [26]: row representation and Markov chain representation. Row representation works by defining the image width and applying a zigzag pattern to keep the bytes adjacent. A Markov chain representation, transmission from one state to another, is achieved using a transmission matrix and a weighted directed graph; each vertex has a value between 0 and 255.

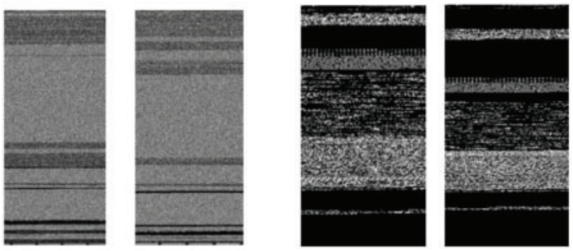
Naeem [20] claimed that color images are more feature enriching than grayscale images. In order to generate the RGB image, you need to combine three matrices for the three channels of an RGB rather than one matrix in a grayscale image. For example, Jian et al. [28] generated the RGB image by combining three matrices, the first channel is a matrix based on binary files, the second channel is a matrix based on a word vector of bytes, and the third channel is a matrix based on word vector of opcodes.

The bulk of malware variants are developed by automation tools or by recycling core function modules; as a result, the binary code or assembly code has a similarity that may be reflected in the images. Malware detection problems are similar to image recognition problems because they both need to identify variants of the original sample. Thus, the method of malware analysis based on visualization technology has it been own unique features and provide a new method for malware detection [28]; Fig. 5 shows the grayscale image of two malware family with variants.

6.2 Virtualization-Based Detection Using Machine Learning

Nataraj et al. [27] proposed a new visualization approach for malware detection based on identifying the malware according to the texture features of grayscale images. They converted malware binaries into grayscale images by converting each Byte into a pixel and extracted features from grayscale images using the GIST algorithm, then building the feature vector and feeding it to the KNN classifier. The experiment claimed an accuracy of 97.18% on the Malimg dataset.

- *Technique*: GIST + KNN
- *Image representation*: Grayscale image
- *Dataset*: Malimg
- *Malware binary*: PE
- *Detecting Malware targeting*: Windows
- *Feature extraction*: GIST
- *Classifier*: KNN
- *Accuracy*: 97.18%



Swizzor.gen!E family [44]   Dontovo.A family [44]

Figure 5: Malware variant image

Makandar et al. [29] presented extracting features using Gabor Wavelet, then passing the texture feature vector to feature selection using PCA to train the classifiers. Two classifiers were used in this experiment, KNN, and SVM, and the performance of each was measured. The experiment used Malimg and Malheur datasets. They used Euclidian, city block, cosine, and correlation with KNN, and Euclidian scored the highest accuracy. The experiment shows KNN with k = 3 scores an accuracy of 98.84% and SVM 98.88%.

- *Technique*: Gabor Wavelet + KNN, Gabor Wavelet + SVM
- *Image representation*: Grayscale image
- *Dataset*: Malimg, Malheur
- *Malware binary*: PE
- Detecting Malware targeting: Windows
- *Feature extraction*: Wavelet transforms
- *Classifier*: KNN, SVM
- *Accuracy*:
    - KNN, k = 3 scores 98.84%
    - SVM scores 98.88%

Hashemi et al. [26] proposed the Local Malicious Pattern (LMP) model, which extracted features using LBP with a KNN classifier. They converted the binary file to a grayscale image using row representation and Markov chain representation. The LBP is used to extract features from the generated image and then feed it into the KNN classifier. The experiment measured the performance of the KNN with different K values. They also converted samples of the malware into assembly code using the Interactive Disassembler (IDA) and compared the performance with state-of-the-art works of opcodes-based detection. They used a different combination of the VX Heaven dataset. The experiment also runs on the Android malware. The proposed method is able to detect Android and Windows malware.

- *Technique*: LBP + KNN
- *Image representation*: Grayscale image
- *Dataset*:
    - Combination of VX Heaven
    - Assembly code from VX Heaven
- *Malware binary*: Assembly Code, Hex dump, PE
- *Detecting Malware targeting*: Android, Windows
- *Feature extraction*: LBP
- *Classifier*: KNN
- *Accuracy*: the model scores 96.20%

Cui et al. [30] established an experiment using deep learning and compared the performance of their model with machine learning state-of-the-art, GIST, and GLCM with KNN and SVM classifiers. GIST and GLCM as shown in Table 2 used for extracting features from the malware image and feed to KNN or SVM for classification. The experiment used a Malimg dataset; these methods can detect the Windows malware. The GIST-KNN scores an accuracy of 91.90%, and the GIST-SVM 92.20%. The GLCM-KNN has an accuracy of 92.50%, and the GLCM-SVM 93.20%.

- *Technique*: GIST + KNN, GIST + SVM
- *Image representation*: Grayscale image
- *Dataset*: Malimg
- *Malware binary*: PE
- *Detecting Malware targeting*: Windows
- *Feature extraction*: GIST
- *Classifier*: KNN, SVM
- *Accuracy*:
    - GIST-KNN scores 91.90%, GIST-SVM scores 92.20%
- *Technique*: GLCM + KNN, GLCM + SVM
- *Image representation*: Grayscale image
- *Dataset*: Malimg
- *Malware binary*: PE
- *Detecting Malware targeting*: Windows
- *Feature extraction*: GLCM
- *Classifier*: KNN, SVM
- *Accuracy*:
    - GLCM-KNN scores 92.50%, GLCM-SVM scores 93.20%

Table 2: A comparative summary of machine learning detection methods

Method	Year	Dataset	Image	Accuracy (%)
Nataraj et al. [27]	2011	Malimg	Grayscale	97.18
Gabor wavelet + KNN [29]	2017	Malimg, Malheur	Grayscale	98.84
Gabor wavelet + SVM [29]	2017	Malimg, Malheur	Grayscale	98.88
LMP-KNN [26]	2018	VX Heaven	Grayscale	96.20
GIST-KNN [30]	2018	Malimg	Grayscale	91.90
GIST-SVM [30]	2018	Malimg	Grayscale	92.20

(Continued)

Table 2: Continued

Method	Year	Dataset	Image	Accuracy (%)
GLCM-KNN [30]	2018	Malimg	Grayscale	92.50
GLCM-SVM [30]	2018	Malimg	Grayscale	93.20

6.3 Virtualization-Based Detection Using Deep Learning

Researchers recently sought to detect malware using deep learning. Building deep learning malware detection classification from scratch is difficult as the malware samples are not significant enough for building and training the model. Furthermore, security researchers use pre-trained deep convolutional neural networks [31–36] models as feature extractors or fine-tuned CNN using transfer learning. Transfer learning has been increasingly used for malware classification. It uses the source domain knowledge to solve the target domain similar type of problem [37,38]. The pre-trained CNN models are trained with millions of images, and it would be easy to use them as a base component in the deep learning-based architecture.

Vasan et al. [39] proposed detecting malware using the IMCEC model with a transfer learning and an ensemble method based on deep neural networks. They suggested using SoftMax and SVM together as classifiers. The feature extraction involves two pre-trained CNN models, VGG16 and ResNet-50. The output of each CNNs is transferred to the SoftMax classifier and multiclass SVMs. The SoftMax classifier and SVM probabilities are combined at a fused step to obtain malware families. The experiment shows 99.50% accuracy for unpacked malware and over 98% for packed malware.

- *Technique*: Transfer learning and fine-tuned CNN
- *Image representation*: Grayscale RGB image
- *Dataset*: Malimg
- *Malware* format: PE
- *Detecting Malware targeting*: Windows
- *Feature extraction*: VGG16 and ResNet-50
- Classifier: SoftMax and Multi-class SVMs
- Result: The model scores an accuracy of 99.50% for unpacked malware and 98% for packed malware

Vasan et al. [40] proposed the IMCFN model as a variant of the previous IMCEC model by converting the malware binary into an RGB image rather than a grayscale image and feeding it to the model. The research compared the proposed method with different pre-trained CNN models VGG-16, ResNet-50, and Google InceptionV3 with the performance of the IMCFN model. The technique achieved overall classification accuracy IMCFN is 98.82% measured, and compared to the other convolutional networks, VGG-16 is 97.12%, RestNet50 is 98.61%, and InceptionV3 is 98.65%.

- *Technique*: Transfer learning and fine-tuned CNN
- *Image representation*: RGB image
- *Dataset*: Malimg and IoT-android
- *Malware format*: PE
- *Detecting Malware targeting*: Windows, Android
- *Feature extraction*: VGG16 and ResNet-50
- *Classifier*: SoftMax and Multi-class SVMs
- *Result*:

  - Malimg with an accuracy of 98.82%
  - IoT-android with an accuracy of 97.35%

Jian et al. [28] proposed the SERLA model, which combines CNN and RNN (Recurrent Neural Network) to improve the accuracy of the classification. They used SEResNet50 and Bi-LSTM with an attention mechanism. The model accepts the RGB images, then pass to SEResNet50, followed by Bi-LSTM, then passes through the attention layer, which is followed by Sigmoid as a classifier. They generated RGB images using by combining three matrices from the binary file, byte word vector, and opcode word vector, then passed them to the augmentation technology CLAHE to augment the three-channel RGB images. The model has an accuracy of 98.31%. Microsoft Big 2015 dataset is used in the experiment.

- *Technique*: CNN + RNN
- *Image representation*: RGB image
- *Dataset*: Microsoft Big 2015
- *Malware format*: Byte file (hexadecimal of the binary content), Assembly Code
- *Detecting Malware targeting*: Windows
- *Feature extraction*: SEResNet50 + Bi-LSTM
- *Classifier*: Sigmoid
- *Result*: 98.31% accuracy

Kumar et al. [41] proposed the DTMIC model, which uses a pre-trained CNN model VGG16 to perform feature extraction from the images. Extracted features are further fed into a fully-connected layer and then passed to the SoftMax classifier to identify the malware family. The new technique in this research, DTMIC, employs an early-stop regularization method to tackle the problem of overfitting in the model and helps with its convergence. They claimed that few researchers had addressed the issue of overfitting when working with insufficient malware datasets. The researchers compared the performance of the state-of-the-art CNNs like VGG16, VGG19, InceptionV3, and RestNet50 with the SoftMax classifier and DTMIC model, which were 98.39%, and 98.71%, 96.25%, 98.71%, and 98.92%, respectively. The experiment shows that the DTMIC score accuracy of 92.92% on Malimg and 93.19% on Microsoft Big 2015.

- *Technique*: Transfer learning and fine-tuned CNN
- *Image representation*: Grayscale image
- *Dataset*: Malimg, Microsoft Big 2015
- *Malware format*: Byte file (hexadecimal of the binary content), Assembly Code
- *Detecting Malware targeting*: Windows
- *Feature extraction*: VGG16
- *Classifier*: SoftMax
- *Result*:
  - Malimg with an accuracy of 98.92%
  - Microsoft (Big 2015) with an accuracy of 93.19%

Naeem [20] proposed the MD-IIOT model for monitoring and detecting malware for the Industry Internet of Things (IIOT). The model observes the network traffic, analyzes and gives a signal to the administrator of the system. The model collected the Android files of IIOT using a sniffing and monitoring unit; they converted the Android .apk files to Java classes files and extracted the byte code, then generated the RGB image from the Java classes. Custom CNNs used to extract features with

ReLU as a classifier. They used Leopard mobile and Malimg datasets with an accuracy of 97.81% and 98.47%, respectively.

- *Technique*: Transfer learning and fine-tuned CNN
- *Image representation*: Grayscale image
- *Dataset*: Leopard mobile, Malimg
- *Malware format*: DEX, PE
- *Detecting Malware targeting*: Android, Windows
- *Feature extraction*: Custom CNN
- *Classifier*: ReLU
- *Accuracy*:
    - Model scores on Leopard mobile datasets 97.81%, and on Malimg dataset scores 98.47%

Bensaoud et al. [42] have introduced the MTL (Multi-Task Learning) model, which attempts to classify a variety of malware binaries, including PE, assembly code, macOS, ELF, IPA, and APK, rather than just PE malware binaries as in previous studies. The image representation RGB image in the BMP and PNG formats is used; these two formats were chosen because they compress the image, thereby reducing the size of the malware image. The model accepts malware images from seven distinct datasets and then passes them to the shared hidden layer, consisting of five convolutional layers followed by two fully-connected layers with PReLU activation. The method does not use any pre-trained CNN models. MTL has seven tasks that help classify malware images on Windows, Linux, macOS, Android, and iOS. They used CycleGAN, one of the Generative Adversarial Networks (GAN) [43–45], to make more malware sample images of Mach-o malware. They tested and compared the accuracy of MTL with PReLU, LeakyReLU, ELU, and ReLU, which were 99.97%, 99.91%, 98.57%, and 93.61%, respectively. The experiment demonstrates that the PReLU classifier is more accurate than other classifiers.

- *Technique*: Multi-task learner with custom CNN
- *Image representation*: RGB image
- *Dataset*:
    - PE malware and benign color image dataset
    - Mach-o adversarial malware and benign color image dataset
    - ELF malware and benign color image dataset
    - Assembly code from PE, ELF, Mach-o, and APK malware and benign color image dataset
    - APK malware and benign color image dataset
    - Mach-o and iOS malware and benign color image dataset
    - Malimg grayscale image dataset
- *Malware format*: PE, Assembly Code, Mach-o, ELF, IPA, APK
- *Detecting Malware targeting*: Windows, Linux, macOS, IOS, and Android
- *Feature extraction*: Custom CNN
- *Classifier*: PReLU
- *Result*: 99.97% accuracy

7 Visualization-Based Detection Using Hybrid Approaches

Virtualization-based malware detection uses images; various image processing techniques could be used with machine learning and deep learning approaches to overcome limitations and improve the model precision. This section emphasizes the possibility of combining machine learning and deep learning techniques. For example, Vasan et al. [39,40] employed two convolution neural networks with SoftMax and multiclass SVMs. Another scenario is the use of CNN with the KNN classifier. Table 3 is a comparative summary of deep learning detection methods.

Table 3: A comparative summary of deep learning detection methods

Method	Year	Dataset	Image	Accuracy (%)
MD-IIOT [20]	2019	Leopard mobile, Malimg	RGB	97.81, 98.47
IMCEC [39]	2020	Malimg	Grayscale	99.50
IMCFN [40]	2020	Malimg	RGB	98.82
SERLA [28]	2021	Microsoft Big 2015	RGB	98.31
DMTIC [41]	2022	Malimg, Microsoft Big 2015	Grayscale	98.92
MTL [42]	2022	Malimg, Generated images	RGB	99.97

8 Conclusion

Traditional anti-malware software is unable to detect unknown malware. A new technique is essential to detect the variant of malware. Visualization-based malware detection recently gained popularity due to accuracy and less time consumption for detecting malware based on the image representation of the malware binary without requiring any domain experts. This paper has presented a detailed review of the current technical status of malware, including malware analysis techniques and malware detection approaches, and focused on virtualization-based detection. It describes different detection techniques based on virtualization.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Norton Labs, "July consumer cyber safety pulse report," 2021. [Online]. Available: https://www.nortonlifelock.com/blogs/norton-labs/july-2021-consumer-cyber-safety-pulse-report.

[2] McAfee, "McAfee labs threat report," 2021. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-threats-jun-2021.pdf.

[3] J. Landage and M. Wankhade, "Malware and malware detection techniques: A survey," *International Journal of Engineering Research*, vol. 2, no. 12, pp. 61–68, 2013.

[4] R. Sihwail, K. Omar and K. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 4–2, pp. 1662–1671, 2018.

[5] Kaspersky, "Ransomware attacks and types–How encryption trojans differ," 2004. [Online]. Available: https://www.kaspersky.com/resource-center/threats/ransomware-attacks-and-types.

[6] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM Comput. Commun.*, vol. 34, no. 2, pp. 39–53, 2004.

[7] G. A. N. Mohamed and N. B. Ithnin, "Survey on representation techniques for malware detection system," *Am. J. Appl. Sci.*, no. 11, pp. 1049–1069, 2017.

[8] S. Talukder, "Tools and techniques for malware detection and analysis," arXiv preprint arXiv:2002.06819, 2020.

[9] S. Talukder and Z. Talukder, “A survey on malware detection and analysis tools,” *International Journal of Network Security & Its Applications (IJNSA)*, vol. 12, pp. 11–12, 2020.

[10] A. Bhardwaj and S. Goundar, “Keyloggers: Silent cyber security weapons,” *Network Security*, vol. 2020, no. 2, pp. 14–19, 2020.

[11] E. Chien and P. Ször, “Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses,” *Virus*, vol. 1, pp. 17–19, 2002.

[12] Cloudmanaged, “Cyber security threat trends: Phishing, crypto top the list,” 2021. [Online]. Available: https://cloudmanaged.ca/wp-content/uploads/2021/09/2021-cyber-security-threat-trends-phishing-crypto-top-the-list.pdf.

[13] A. Zimba, Z. Wang, M. Mulenga and N. H. Odongo, “Crypto mining attacks in information systems: An emerging threat to cyber security,” *Journal of Computer Information Systems*, vol. 60, no. 4, pp. 297–308, 2020.

[14] D. Yaga, P. Mell, N. Roby and K. Scarfone, “Blockchain technology overview,” arXiv preprint arXiv:1906.11078, 2019.

[15] Norton, “What is adware,” 2020. [Online]. Available: https://us.norton.com/internetsecurity-emerging-threats-what-is-grayware-adware-and-madware.html.

[16] R. Tahir, “A study on malware and malware detection techniques,” *International Journal of Education and Management Engineering*, vol. 8, no. 2, pp. 20, 2018.

[17] Norton, “Malvertising: What is it and how to avoid it,” 2020. [Online]. Available: https://us.norton.com/internetsecurity-malware-malvertising.html.

[18] Finance Yahoo, “Bitcoin miners earn record hourly revenue of $4M,” 2021. [Online]. Available: https://finance.yahoo.com/news/bitcoin-miners-earn-record-hourly-102208827.html.

[19] M. H. Khan, I. R. Khan and others, “Malware detection and analysis,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 653–658, 2017.

[20] H. Naeem, “Malware detection in industrial internet of things based on hybrid image visualization and deep learning model,” *Ad Hoc Networks*, vol. 105, pp. 102154, 2020.

[21] I. A. Saeed, A. Selamat and A. M. Abuagoub, “A survey on malware and malware detection systems,” *International Journal of Computer Applications*, vol. 67, no. 16, pp. 120–124, 2013.

[22] I. You and K. Yim, “Malware obfuscation techniques: A brief survey,” in *Proc. of the Int. Conf. on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, Fukuoka Prefecture Japan, pp. 297–300, 2010.

[23] J. Singh and J. Singh, “Challenge of malware analysis: Malware obfuscation techniques,” *International Journal of Information Security Science*, vol. 7, no. 3, pp. 100–110, 2018.

[24] I. Yoo, “Visualizing windows executable viruses using self-organizing maps,” in *Proc. of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, America, pp. 82–89, 2004.

[25] G. Conti, E. Dean, M. Sinda and B. Sangster, “Visual reverse engineering of binary and data files,” in *Int. Workshop on Visualization for Computer Security*, America, pp. 1–17, 2008.

[26] H. Hashemi and A. Hamzeh, “Visual malware detection using local malicious pattern,” *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 1–14, 2019.

[27] L. Nataraj, S. Karthikeyan, G. Jacob and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in *Proc. of the 8th Int. Symp. on Visualization for Cyber Security*, America, pp. 1–7, 2011.

[28] Y. Jian, H. Kuang, C. Ren, Z. Ma and H. Wang, “A novel framework for image-based malware detection with a deep neural network,” *Computers & Security*, vol. 109, pp. 102400, 2021.

[29] A. Makandar and A. Patrot, “Malware class recognition using image processing techniques,” in *Proc. of the Int. Conf. on Data Management, Analytics and Innovation (ICDMAI)*, Pune, India, pp. 76–80, 2017.

[30] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang *et al.,* “Detection of malicious code variants based on deep learning,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[31] N. Aloysius and M. Geetha, “A review on deep convolutional neural networks,” in *Proc. of the Int. Conf. on Communication and Signal Processing (ICCSP)*, Chennai, India, pp. 0588–0592, 2017.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[33] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 15–18, 2021.

[34] Viso, "VGG very deep convolutional networks (VGGNet)–what you need to know," 2014. [Online]. Available: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.

[35] M. Jogin, M. Madhulika, G. Divya, R. Meghana, S. Apoorva *et al.,* "Feature extraction using convolution neural networks (CNN) and deep learning," in *83rd IEEE Int. Conf. on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, pp. 2319–2323, 2018.

[36] O. Russakovsky, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] F. Zhuang, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[38] S. Tammina, "Transfer learning using VGG-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, pp. 143–150, 2019.

[39] D. Vasan, M. Alazab, S. Wassan, B. Safaei and Q. Zheng, "Image-based malware classification using ensemble of CNN architectures (IMCEC)," *Computers & Security*, vol. 92, pp. 101748, 2020.

[40] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei *et al.,* "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, pp. 107138, 2020.

[41] S. Kumar and B. Janet, "DTMIC: Deep transfer learning for malware image classification," *Journal of Information Security and Applications*, vol. 64, pp. 103063, 2022.

[42] A. Bensaoud and J. Kalita, "Deep multi-task learning for malware image classification," *Journal of Information Security and Applications*, vol. 64, pp. 103057, 2022.

[43] I. Goodfellow, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, pp. 23–37, 2014.

[44] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta *et al.,* "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[45] A. Aggarwal, M. Mittal and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, pp. 100004, 2021.