Firat Univ Jour. of Exp. and Comp. Eng., 2(2), 55-68, 2023

Firat University Journal of Experimental and Computational Engineering
Fırat Üniversitesi Deneysel ve Hesaplamalı Mühendislik Dergisi

A YARA-based approach for detecting cyber security attack types

Siber güvenlik saldırı türlerini tespit etmek için YARA tabanlı bir yaklaşım

Kubra YILDIRIM [1*], Mustafa Emre DEMIR[2], Tugce KELES[3], Arif Metahan YILDIZ[4], Sengul DOGAN[5], Turker TUNCER[6]

[1,2,3,4,5,6]Department of Digital Forensics Engineering, College of Technology, Firat University, Elazig, Turkey.
[1]kubra.yildirim@firat.edu.tr, [2]mustafaemredemir@gmail.com, [3]tkeles@firat.edu.tr, [4]a.metehanyildiz@gmail.com, [5]sdogan@firat.edu.tr, [6]turkertuncer@firat.edu.tr

Received: 01.03.2023
Accepted: 10.05.2023
Revision: 22.03.2023
doi: 10.5505/fujece.2023.09709
Research Article

Abstract
Technological advancements have recently propelled individuals, institutions, and organizations to conduct their business processes on information systems. However, keeping personal and corporate data on information systems has given rise to issues related to data security. The accessibility of data on information systems has made it vulnerable to theft and exploitation by malicious groups or individuals, thus posing a significant risk to data security. Consequently, the demand for data security has led to a new business sector offering various cybersecurity solutions to protect organizations' systems. This paper presents an analysis of the prevalent types of cyber attacks worldwide. The study aims to create a virtual environment with Windows and Linux systems in Forensic Informatics and Incident Response processes to apply frequently used cyber attack methods, develop defense mechanisms against these methods, and contribute to revealing the root cause by solving the incident pattern. Furthermore, this application demonstrates how manual techniques and open-source solutions, such as YARA, can be used to detect malware derivatives commonly found in Windows systems.

Keywords: YARA, Malware, Digital forensics, Cyber attack

Özet
Teknolojik gelişmeler son dönemde kişi, kurum ve kuruluşları iş süreçlerini bilgi sistemleri üzerinde yürütmeye yöneltmiştir. Ancak kişisel ve kurumsal verilerin bilgi sistemleri üzerinde tutulması veri güvenliği ile ilgili sorunları gündeme getirmiştir. Bilgi sistemlerindeki verilerin erişilebilirliği, onu hırsızlığa ve kötü niyetli gruplar veya kişiler tarafından sömürülmeye karşı savunmasız hale getirdi ve bu nedenle veri güvenliği için önemli bir risk oluşturdu. Sonuç olarak, veri güvenliğine yönelik talep, kuruluşların sistemlerini korumak için çeşitli siber güvenlik çözümleri sunan yeni bir iş sektörüne yol açmıştır. Bu makale, dünya çapında yaygın olan siber saldırı türlerinin bir analizini sunmaktadır. Çalışmada, Adli Bilişim ve Olay Müdahale süreçlerinde Windows ve Linux sistemleri ile sanal ortam oluşturularak sıklıkla kullanılan siber saldırı yöntemlerinin uygulanması, bu yöntemlere karşı savunma mekanizmalarının geliştirilmesi ve olay örüntüsünün çözülerek kök nedenin ortaya çıkarılmasına katkı sağlanması amaçlanmaktadır. Ayrıca bu uygulama, Windows sistemlerinde yaygın olarak bulunan kötü amaçlı yazılım türevlerini tespit etmek için manuel tekniklerin ve YARA gibi açık kaynaklı çözümlerin nasıl kullanılabileceğini gösterir.

Anahtar kelimeler: YARA, Zararlı yazılım, Adli bilişim, Siber saldırı

1. Introduction

Cyberattacks involve the intentional actions of individuals or groups who use one or more computer systems to target organizations' computers or network systems that store data and information [1]. These attacks aim to achieve various goals, including disrupting the normal functioning of systems, stealing confidential data, or exploiting compromised computers to launch further attacks [2]. Cybercriminals use different tactics to carry out cyber attacks, such as denial of service attacks, phishing attacks, and malware attacks [3]. However, the primary objective of cyber attackers is to exploit vulnerabilities in physical or logical resources, thereby compromising the confidentiality, integrity, or accessibility of the targeted resource [1].

*Corresponding author

In today's digital landscape, cyber attack groups have become ubiquitous, emphasizing the need to develop new techniques for incident response to analyze such attacks [4, 5]. The incident response involves detecting existing threat elements in all systems after an institution or organization is exposed to Advanced Persistent Threats (APTs). In addition, it involves identifying the incident's root cause, removing persistent threat elements, ensuring they do not recur, and securing the systems again. These processes ensure timely delivery [6], and the main objective of incident response is to swiftly eliminate threat elements and ensure the security of the systems once again [7].

Digital forensics and incident response (DFIR) is a highly specialized field that focuses on identifying, mitigating, and investigating cybersecurity incidents. It involves gathering, preserving, analyzing, and presenting forensic evidence to establish an in-depth understanding of the events. Furthermore, DFIR strives to control, halt, and prevent attacks while restoring systems by minimizing any damage incurred [8, 9].

The following studies on YARA-related cybersecurity are presented in the literature. Kim et al. [10] developed an automated, low-interaction malicious web page detector called WebMon that utilizes machine learning and YARA signatures against Web browser threats to detect invasive roots in Web resources loaded from WebKit2-based browsers. Kumar et al. [11] investigated WannaCry intrusion and possible ransomware detection using the YARA rule-based detection technique. Rosyid et al. [12] utilized a Honeypot as a security sensor to detect malware and attempted to classify the type of malware found by scanning suspicious files with YARA. Siddabathula et al. [13] employed YARA for robust rule-based detection to intercept packets thrown into the network to cause an attack. They also proposed an automated system for analyzing these packages using YARA. Si et al. [14] proposed a malware detection method based on automatic YARA rule generation on dynamic behaviors to enhance malware detection in terms of automation and efficiency. Naik et al. [15] suggested that fuzzy rules be included in YARA rules to optimize performance during execution since YARA rule conditions are primarily Boolean expressions that focus on the binary outcome of malware analysis and can restrict the optimized use of YARA rules and findings. Khalid et al. [16] developed a framework that automates generating high-quality, effective, and efficient malware signatures with less time and effort. Their approach provides a general strategy for automatic YARA rule-based signature generation by carefully selecting the most promising key ideas of the relevant work. Testing the prototype demonstrated that it can detect samples with an average accuracy of 95.00%. Xu et al. [17] introduced the Bert-based strings language model (beslm), which can effectively learn and highly filter the semantic information of YARA rules and the association relationship between strings through the pre-training and fine-tuning phase. Experimental results indicated that the YARA rules automatically generated by beslm improved an average of 13.3%, 13.8%, and 15.1% across the three detection metrics compared to the comparison method. Whether rules are created manually or automatically, it is desirable to improve both the process performance and the detection results. Naik et al. [18] proposed a method that leverages fuzzy hashing and fuzzy rules techniques to increase the efficiency of YARA rules while minimizing their complexity and overhead. The method involves generating fuzzy hashes, referred to as "fuzzy YARA rules" in their work. Naik et al. [19] evaluated the effectiveness of automatically generated YARA rules using various Python-based open-source tools, such as yarGen, YaraGenerator, and yabin. The authors proposed a method to improve the automatically generated YARA rules by incorporating a fuzzy hash method, which significantly increased the effectiveness of YARA rules regardless of the tool used to generate them. This was demonstrated through various experiments on collected malware and good software samples. Raff et al. [20] utilized large n-grams ($n \geq 8$) combined with a new binary clustering algorithm to generate simple YARA rules more efficiently than currently available software. Bilstein et al. [21] presented YARA-Signator, an approach for the automatic generation of code-based YARA rules based on the isolation of instruction n-grams that are common in a particular malware family but not found in any other family. By applying YARA-Signator to the Malpedia dataset, the authors achieved an overall F1 score of 98.30%, resulting in only a few false positives in a sanity check against a large good software dataset. Web application source code security can be enhanced by detecting malicious code such as web shells. Nguyen et al. [22] proposed a deep learning approach for detecting and identifying malicious code in PHP source files using pattern matching techniques and YARA rules to generate malicious and harmless datasets. The authors converted PHP source codes into a numerical sequence of PHP opcodes and used a Convolutional Neural Network model to make a prediction on whether malicious code such as a webshell was present in the PHP file. The proposed method achieved a high accuracy rate of 99.02% with a low false positive rate of 0.85%.

1.1. Objective and motivation

The motivation for this study is given as follows. With the proliferation of technology and the consequent upsurge in digital data, the significance of data security has become increasingly apparent. Data storage in information systems creates an easy avenue for malevolent entities to access and steal data, thus rendering data security a top priority for organizations. Notably, organizations have resorted to various cyber security measures to counter cyber attacks, resulting in a new business sector in the field. This study aims to establish a virtual environment comprising Windows and Linux systems to execute commonly applied cyber attack methods and devise corresponding defense mechanisms. Concomitantly, application processes are highlighted to contribute to the identification of the underlying causes and evaluate methods that could be leveraged in digital forensics and incident response processes. In essence, this study endeavors to raise awareness of data security in the information technology domain and impart knowledge to interested parties. Moreover, this work aims to proffer potential approaches for safeguarding organizational data. Thus, this paper could serve as a vital source of motivation for professionals working in data security.

2. Cyber Attacks

A cyber attack is an intentional action aimed at compromising computers or computer networks in order to steal, alter, or damage important data. Cyber attacks can be classified into different categories based on the techniques employed by the attacker. Some cyber attacks in the literature are given below.

Distributed Denial of Service Attack (DDoS): A Distributed Denial of Service (DDoS) attack is a form of cyber attack that overwhelms the targeted information system, server, or network by flooding it with a massive volume of internet traffic, causing it to become unresponsive. This is accomplished by exploiting previously compromised systems such as servers, computers, telephones, and IoT devices connected to the network [23]. In contrast, a Denial of Service (DoS) attack is a type of cyber attack where a single source tries to flood a target with traffic to consume its resources and make it inaccessible.

The most prevalent way to identify a DDoS attack is by noticing a sudden slowdown or inaccessibility of a service or resource. In such cases, detailed analysis and research are necessary to investigate the root cause of the problem. This type of research requires specialized tools to monitor and analyze network traffic. The following information is essential for detecting DDoS attacks using such analysis tools [24].

- Controlling suspicious traffic from an IP range or a single IP is necessary.
- Depending on the attack type, there may be various relics to detect DDOS attacks.
- A single page or a single service does not work on the serviced system, and there is a sudden increase in requests made to these areas.
- A sudden increase in requests for the service or a continuation of the increase at certain time intervals may occur.
- Traffic flow may occur where technical information about location, device type, or web browser can be detected.

To launch DDoS attacks, computers need to be connected to a network. Some devices in these connected networks are vulnerable to malware and unauthorized access, and attackers can remotely control these devices. Such compromised devices are referred to as bot machines. Groups of bot machines are known as botnet networks or botnet groups.

The botnet networks created by attackers are designed to be controllable. An attacker can remotely send commands to each device in the botnet network and redirect them for a planned attack. The attacker sends requests to a target network, device, or service from the botnet network they created, potentially causing network or server overload. This situation can make normal traffic inaccessible, and the service may be blocked due to the attack. Since each bot in the botnet is a legitimate internet-connected device, it can be challenging to differentiate between normal traffic and attacker traffic in some cases [25].

Phishing Attacks: The technique of phishing involves presenting oneself as a legitimate service, company, or individual to a targeted person or community and building trust. The objective is to obtain sensitive data such as the victim's personal information, credit card details, or passwords. This technique is typically employed via e-mail, text messages, or phone calls. The information obtained through such attacks is then used to gain access to crucial accounts and can lead to identity theft or financial losses [26]. The first case of phishing was filed in 2004 in California against an individual who created a fake America Online website to obtain sensitive user information, including credit card details for fraudulent purposes. In addition to phishing, attackers employ other techniques such as 'vishing' (voice phishing), 'smishing' (SMS phishing), and several other methods to deceive victims [27].

Malware Attacks: Malware attacks are a type of cyber attack where malicious software is designed to damage or monitor a computer, server, client, computer network, or network without the knowledge of the end user.
Online criminals frequently use malware to steal customer, financial, or corporate information. While their motivations may differ, cyber attackers often focus their tactics, techniques, and procedures (TTP) on gaining access to more powerful credentials and accounts to carry out their objectives [28].

Most types of malware can be classified into the following categories [29];

- Active types of cyber attacks: Virus, Worm, trojan, Spyware, Brute force attacks, Cross-site scripting (XSS), etc.
- Types of Passive Cyber Attacks: Computer surveillance, Network surveillance, Wiretapping, Data scraping, Typosquatting, Keystroke monitoring (keylogging)
- Types of Industrial Cyber Attacks: Attacks on electricity networks, Attacks on natural gas lines, Attacks on financial infrastructures, Attacks on telecommunication systems, Attacks on transportation infrastructure, Attacks on water infrastructure.

Zero-Day Attack: A zero-day attack is a type of cyber attack that exploits an unknown vulnerability in an application or operating system of an information system. As the vulnerability is unknown, these attacks often occur without the users' knowledge. Mitigating zero-day attacks is crucial to designing secure and effective applications [30].
The salient features of zero-day attacks are [31]:

- Zero-day attacks typically occur when the vulnerability is first discovered and exploited, and the application developers release the necessary solution to counter the exploit. This timeline is often referred to as the vulnerability window.
- Zero-day attacks can render a network unusable by exploiting the vulnerabilities of related applications.
- They are not always viruses and can take on other forms of malware, such as Trojan horses or worms.
- For PC users, it is extremely difficult to diagnose a zero-day attack as the nature of the attack is through a trusted entity.
- Using the latest anti-malware solutions is the most effective, although it can provide only minimal security against a zero-day attack.

Man-in-the-Middle Attack: Man-in-the-Middle (MiTM) attacks are a malicious technique where an attacker secretly intercepts and manipulates communication between two parties who believe they are communicating directly. This type of cyberattack enables eavesdropping and even full control of the entire conversation, putting sensitive personal data such as login credentials, account information, or credit card numbers at risk [32]. MiTM attacks go by different names such as man-in-browser, machine-in-the-middle, monkey-in-the-middle, and monster-in-the-middle attacks. The most common type of MiTM attack targets the victim's web browser, usually by infecting their device with malware, which is often distributed through phishing emails. These attacks aim primarily to steal financial information by preventing the user from accessing banking or financial websites [33].

During a MiTM attack, cybercriminals position themselves in the middle of online communication or data processing, allowing them to access the user's web browser and the data transmitted and received during online transactions by spreading a virus. Online banking and e-commerce platforms that require secure authentication using public and private keys are the primary targets of MiTM attacks, as they allow attackers to obtain login passwords and other sensitive information. These attacks typically involve a two-step process known as data capture and decryption. First, the attacker must eavesdrop on the data transfers between the client and the server to intercept the data. By deceiving the client and server into thinking they are exchanging information, the attacker can intercept the data, connect to the legitimate site, and act as a proxy to read and modify the conversation [34].

3. Malware Detection with YARA

In a forensic investigation, a crucial aspect is identifying and classifying potentially malicious executables in a system or network [35]. Malware detection is typically done by identifying known features of files. To detect such malware, the hash value of the suspicious file is calculated and compared with the hash values of known malware. However, this method is not foolproof since pests can evade detection. This is because the hash values of malware can be easily changed. Hash function algorithms produce a constant output, regardless of the input. As a result, a slight alteration in the input will completely change the output. In this way, it becomes easy to alter the hash values of malicious software. For instance, changing the name of a parameter in the malicious code or adding a comment line will entirely alter the hash value of the malware and keep it hidden [36].

Antivirus software and enterprise malware detection solutions use various techniques to identify and classify malware. One approach is to scan a specific text set in a file that identifies certain types and family groups of malware. Another method is to look for a series of bytes specific to a particular malware. Additionally, some solutions employ behavioral analysis to detect malware based on its actions, ranging from basic changes to advanced behavioral patterns in computer systems.

Malware is software intended to harm computer systems, including stealing data and identity, espionage, and providing its developer full or limited control [37]. YARA is an open-source tool that uses a rule-based approach to identify malware based on signature detection, such as text or binary patterns. Rules or descriptions are created from strings and logic, and they match patterns or features to classify the sample according to specific malware families or variants [38]. YARA rules are a practical tool for cybersecurity analysts to use in any field. However, developing high-quality YARA rules to detect a particular malware family can be time-consuming and challenging, even for experts in the field. YARA rules categorize and identify malware samples by creating malware family descriptions based on textual or binary patterns. Text or binary patterns can be constructed to match a file or part of a file using YARA rules to discover dangerous files quickly. YARA uses a rules file in the "yar" or "YARA" format, where it scans a file system to identify any file that meets the criteria defined by the rules. YARA was developed by VirusTotal and made available free of charge.

In the simplest terms, a YARA rule is given below.

```
rule pseudocode_yara : example

{
      meta:
            description = “This is an example YARA rule.”
            threat_level = 3
      string:
          $a = { 6A 40 68 00 30 00 00 6A 14 8D 91}
          $c = "malware"
      condition:
            $a or $b
}
```

The example rule named pseudocode _yara given above and starting with "rule" can be defined as the title indicating the beginning of a YARA rule. The string sozdekod_yara is the chosen name for the YARA rule. For the YARA rule to be successful, choosing a suitable title for its content is important. The "meta:" section is the area with the details and metadata about the rule. The information here is important for defining the rule. The "strings" section is where data can be written in any supported format and desired to be determined about the rule. The conditions you want to regulate so that your YARA rules can cause a match are specified in the "condition:" or condition section. All conditional terms can be applied here [39]. YARA is a set of rules for detecting malicious files in any computer environment. It facilitates the detection of malicious activities. It works fast and is platform-independent. The installation does not require. For using YARA, the tool is downloaded from the address "http://virustotal.github.io/yara/" and scanned by running the command string "Yara.exe –mrs <rule.yar> <_directory to scan>".

3.1. Example of using YARA

Digital Forensics is a branch of forensic science that includes digital technology. The ultimate goal of digital forensics is to collect and preserve evidence that will help prosecute cybercrimes should the criminals behind an attack face criminal charges. Cybersecurity analysts focus on acquiring, researching, and examining data in digital environments. For example, you work in the CERT team as a cybersecurity analyst and obtain a malicious file on one of your systems. Your manager commissioned you to analyze the file, collect unique data, write a YARA rule, and search it across the organization. Let's say the malicious file is an executable ".exe" file but disguises itself as a ".pdf" file to fool users, thus making those running the malware think of it as a regular pdf file. You detect that the name of this file is "maas-zam". You need to review this and check for the existence of this file across the entire institution. In this example, we will explain how to do this.

We can calculate the Hash of this file using various tools and check if it exists on platforms like VirusTotal. After calculating the hash of the file, we need to parse the unique strings in the malicious file to generate the YARA rule. Apart from static analysis techniques, the "string.exe" tool developed by Microsoft can be used for this. This tool extracts all string values in the file. Then all that remains is to separate specific areas for the file with visual inspection. The output of this example file provided by the "string.exe" tool is shown in Figure 1.

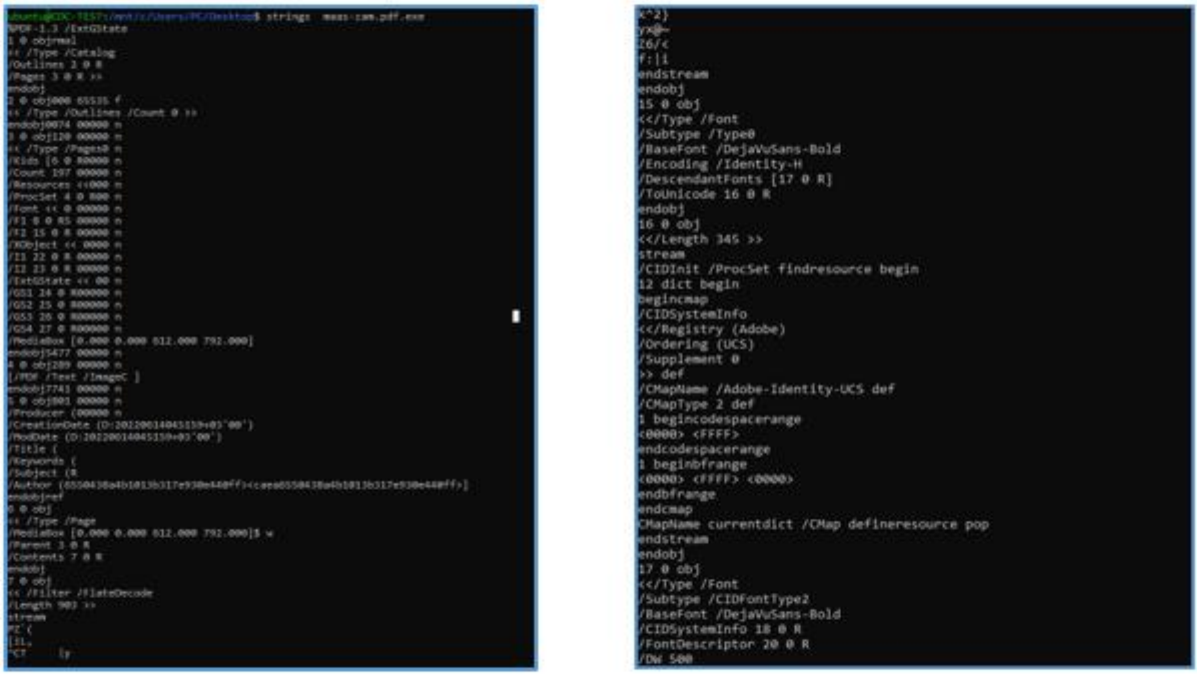

Figure 1. Malware String Output

Among the string values obtained in Figure 1, values that may be unique are parsed and used to write rules. In the simplest way, a YARA rule can be written like this..

4. The Attack Application and Results

4.1. An example attack application

In order to simulate attack types, a virtual environment with Windows and Linux systems has been created. These types of attacks were applied to this environment in order to set an example for the attacks mentioned. The outputs of the application are presented under this title.

4.2. Phishing attack example

This study was conducted to simulate phishing attacks, the most common cyber attack technique organizations encounter, and to steal the saved passwords on the user's browser. In this direction, the e-mail address of a legal institution was imitated, and an e-mail was sent to the user's device in the virtual environment. According to the scenario, when the user created in the virtual environment clicks on the incoming e-mail, the malware downloads from the "discord" channel to the user's device. The aim is to upload the malware and download it from the open "discord" channel. When the user runs the downloaded malware, the password saved in the user's browser and the password saved in the e-mail box is sent to the system previously provided by the attackers via the web channel. It aims to transfer the data here by using the file transfer protocol over an IP address.

When the user opens the malicious file in the attached e-mail attachment, the malware will be registered on the user's computer and will run automatically. The purpose of this scenario is to show the effects of a phishing attack. Here, the analysis of the persistent malware will be explained first, and then the YARA rule will be created for detection.

The malware is compiled as 32-bit and has a Windows interface. There are "autoit" codes in the application. AutoIt is a scripting language used for automation, developed for the Windows interface and general scripts. We see that the application used in the simulation is not packaged. This means that analyzing the malware is difficult because the techniques used are not used in this malware. Such malware is easier to examine and analyze. The class of malware is selected as "dropper". Dropper is a type of trojan designed to "install" malware (virus, backdoor, etc.) on the target system. The malware code is designed in a single phase so that virus scanners do not detect it. Once activated, it can download the malware to the target machine. Detailed information about the malware is shown in Figure 2.

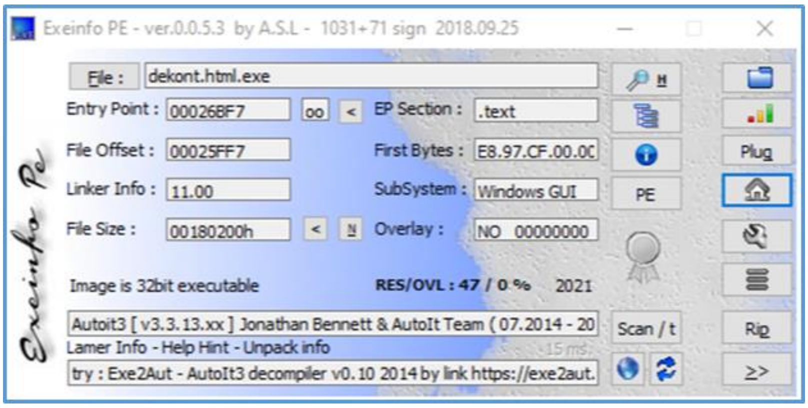

Figure 2. Learning Details of the Executable File

The application is compiled in C++. The compilation time is 28 July 2021. Using the Pestudio tool, we can access much information about the malware. 5 out of 44 indicators that may indicate that the application is harmfully matched this application. Among these, virus total results, suspicious libraries used by the application, and tactics used in a platform used to classify malicious software called "mitre framework" are seen to match. Details of this information are shown in Figure 3.

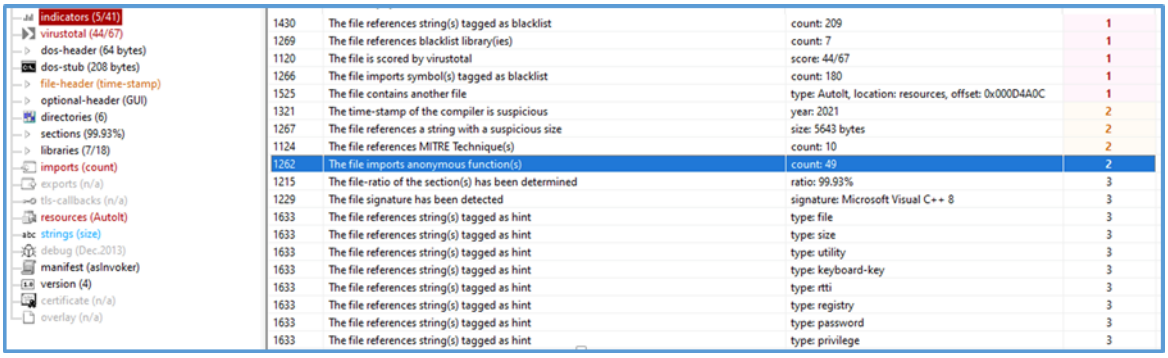

Figure 3. Malware Investigation Details

String expressions in the application also provide information about the malware. The most important factor in creating the YARA rule, which forms the basis of our study, is obtaining the unique string values contained in the malware and belonging to that pest. The malware we use here does not have a certificate. The string expressions in the application show that it runs some processes, changes registry values, and tries to gather information about the system.

By using the AutoItExtractor application, the code block in the malware can be viewed. Here, the codes of the malware are displayed using the relevant application. Code complexing technique was used to make the analysis of the malware difficult. This technique replaces variable names and values with random expressions, making it difficult for the examining analyst to learn the malware's main purpose. Malware analysis can take time when the codes are complicated here. For this reason, as a result of the examination, it is understood that file creation, deletion, and writing operations are performed. The screenshot from which this information is obtained is presented in Figure 4.

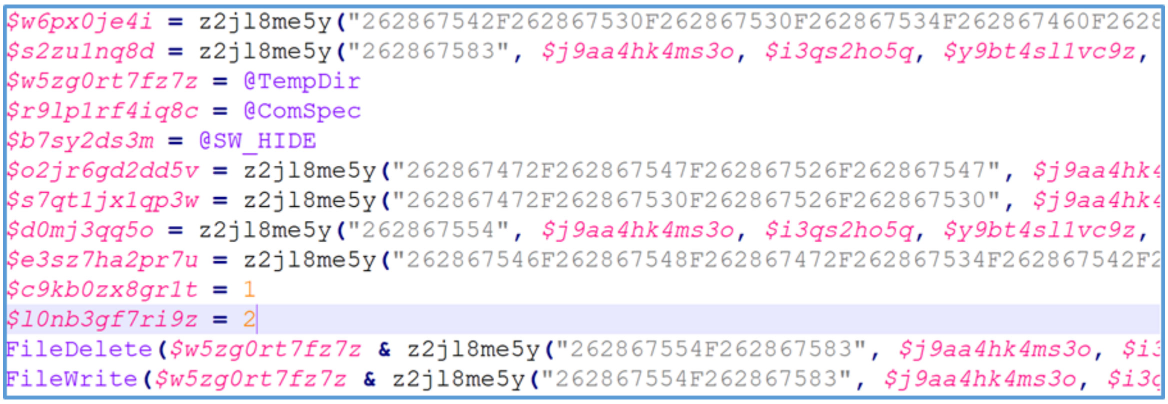

Figure 4. Analysis of Malicious Codes

A function draws attention to the malware code obtained with Autoit. A variable inside the function starts with 4D5A. In the Windows operating system, every executable application starts with a value known as MZ. Extracting a different executable file embedded in the malware indicates that the malware belongs to the "dropper" category. The first-word "MZ" is used for executable files when the file is opened. Its hexadecimal equivalent is “4D5A”. A screenshot of this finding is shown in Figure 5.

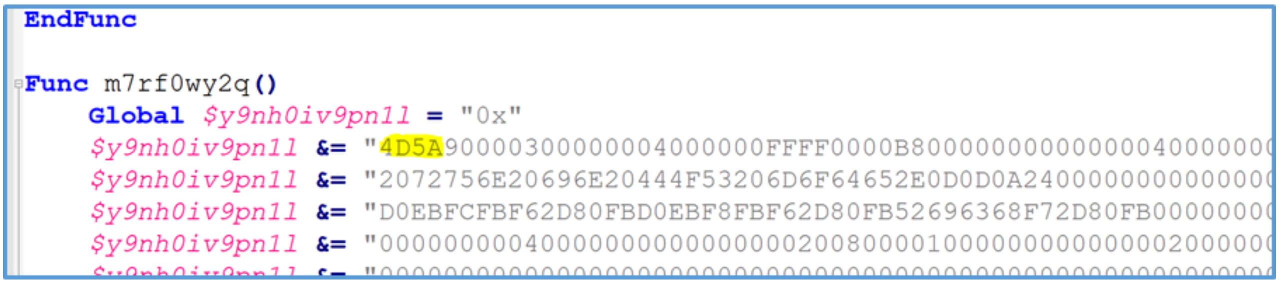

Figure 5. Detecting the Executable File

With the "BinaryToString" function, the embedded executable file is converted from binary format to string format. Then the files are extracted to the "%temp%" folder. The code for this finding is given below.

Return BinaryToString ($y9nh0iv9pn11)

When the application is run, and the strings in the RAM memory are analyzed, a domain name draws attention. It will be understood later that the passwords detected through the system are uploaded to the domain detected here. This domain is legal and still in use. Attackers use previously captured systems to avoid attracting attention for their purposes. This information is masked. Details of this information are shown in Figure 6.

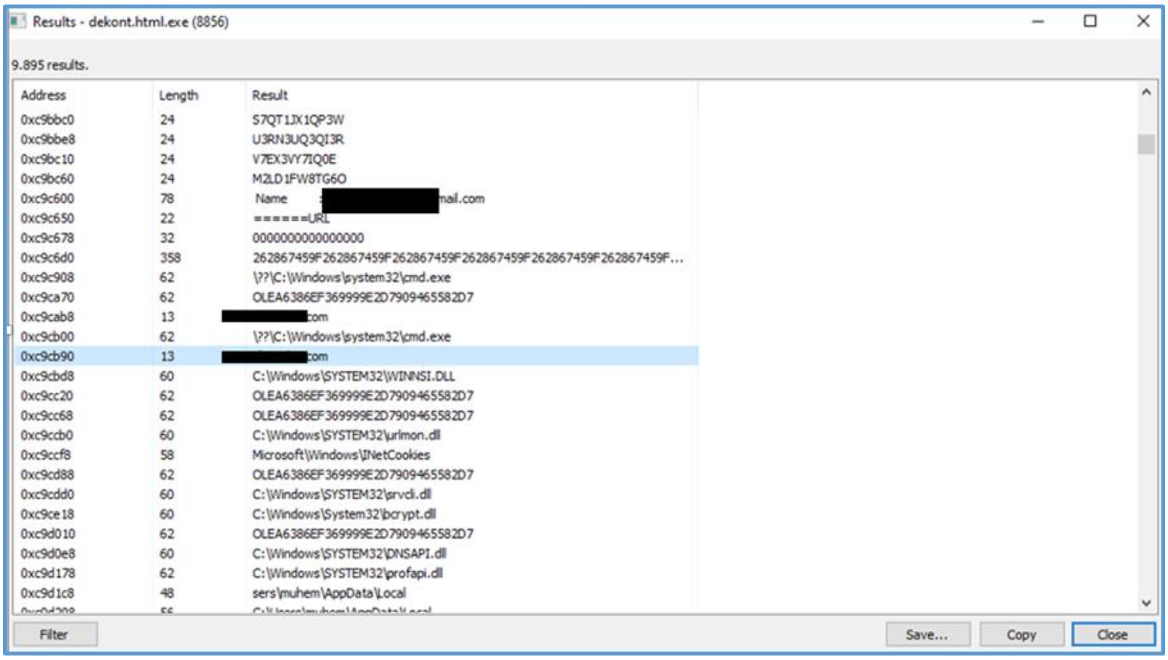

Figure 6. Command and Control Center Detection

It is seen in the temporary memory (memory) that the malware connects to the relevant domain with the "password" parameter. The detail of this information is shown in Figure 7.

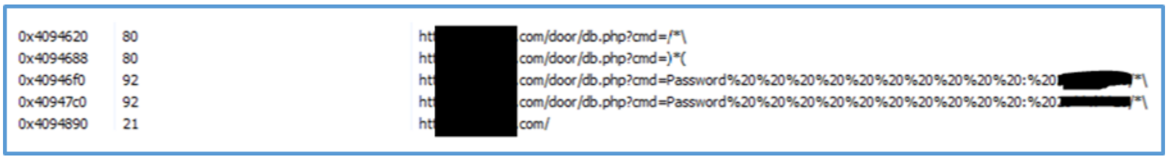

Figure 7. RAM Analysis

As can be seen in the image in Figure 9 below, the malware accesses the "\\AppData\\Local\\Microsoft\\Edge\\User Data\\Default\\Login Data\\" file where the Microsoft Edge browser keeps the saved passwords.

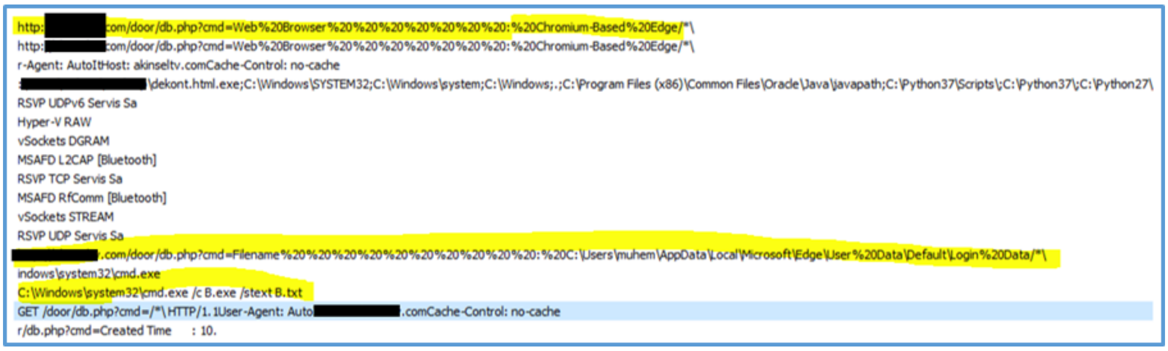

Figure 8. Detection of Malware Target

It has been determined that the command "c:\Windows\system32\cmd.exe /c B.exe /stext B.txt" runs when the malware runs. When the malware runs, two applications named "A.exe" and "B.exe" run under the "temp" directory. At the same time, two text files named "A.txt" and "B.txt" are created under the temp directory.

The malware is to run these 2 applications and write their output to a text file and send it to the command control (C2) address via the web channel. It has been understood that the address used as command and control is a news website. To avoid arousing suspicion, attackers take advantage of the vulnerability on this website and use it as a command and control server. The application named "A.exe" (Web Browser Password Viewer) is an application that allows you to see the saved passwords on the already-used browser. This App is bundled with UPX. UPX(Ultimate Packer for Executables) is an open-source packaging application that supports many file formats by different operating systems. Likewise, the mailbox password viewing application named "B.exe" (E-mail Password Recovery) is also understood to be packaged with UPX. Details of malware are shown in Table 1.

Table 1 A.exe and B.exe detail.s

property	value
md5	62B2864C32CB33F57A65F47269D91BE4
sha1	D072FF4E71B3F53E3D198067A61BCDD835CA0D92
sha256	40257944035022DB1474E714C256585977F8A89D8F960FA040A64567DE67194A
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z …………………………………………………………………………….
file-size	195584(bytes)
size-without-overlay	n/a
entropy	7.874
imphash	n/a
signature	UPX www.upx.sourceforge.net
entry-point	60 BE 00 90 42 00 8D BE 00 80 FD FF 57 EB 0B 90 8A 06 46 88 07 47 01 DB 75 07 88 1E 83 EE FC 11 DB
file-version	1.90
description	E-mail Password-Recovery
file-type	executable

Table 1 A.exe and B.exe detail.s (Contain)	
cpu	32-bit
subsystem	GUI
compiler-stamp	0x5DE03B9B (Fri Nov 29 00:26:51 2019)
debugger-stamp	n/a
resources-stamp	empty
exports-stamp	n/a
version-stamp	empty

(a) A.exe Details

property	value
md5	62B2864C32CB33F57A65F47269D91BE4
sha1	D072FF4E71B3F53E3D198067A61BCDD835CA0D92
sha256	40257944035022DB1474E714C256585977F8A89D8F960FA040A64567DE67194A
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z ……………………………………………………………………………………..
file-size	195584(bytes)
size-without-overlay	n/a
entropy	7.874
imphash	n/a
signature	UPX www.upx.sourceforge.net
entry-point	60 BE 00 90 42 00 8D BE 00 80 FD FF 57 EB 0B 90 8A 06 46 88 07 47 01 DB 75 07 88 1E 83 EE FC 11 DB
file-version	1.90
description	E-mail Password-Recovery
file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	0x5DE03B9B (Fri Nov 29 00:26:51 2019)
debugger-stamp	n/a
resources-stamp	empty
exports-stamp	n/a
version-stamp	empty

(b) B.exe Details

When the malware is run and the network packages are examined, it is understood that it is trying to send the user name and password information to the command and control server similarly. Details of this information are shown in Figure 9.

Protocol	Length	Info
HTTP	299	GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?b8dd0bb4ea07dab0 HTTP/1.1
HTTP	207	GET /door/db.php?cmd===/*\ HTTP/1.1
HTTP	157	GET /door/db.php?cmd=)*(HTTP/1.1
HTTP	247	GET /door/db.php?cmd=URL%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20:%20https://www.virustotal.com/gui/sign-in/*\ HTTP/1.1
HTTP	216	GET /door/db.php?cmd=Web%20Browser%20%20%20%20%20%20%20:%20Chromium-Based%20Edge/*\ HTTP/1.1
HTTP	172	GET /wpad.dat HTTP/1.1
HTTP	184	GET /wpad.dat HTTP/1.1
HTTP	222	GET /door/db.php?cmd=User%20Name%20%20%20%20%20%20%20%20%20:%[redacted]@gmail.com/*\ HTTP/1.1
HTTP	209	GET /door/db.php?cmd=Password%20%20%20%20%20%20%20%20%20%20[redacted]*\ HTTP/1.1
HTTP	189	GET /door/db.php?cmd=Password%20Strength%20:%20Strong/*\ HTTP/1.1
HTTP	194	GET /door/db.php?cmd=User%20Name%20Field%20%20%20:%20email/*\ HTTP/1.1
HTTP	197	GET /door/db.php?cmd=Password%20Field%20%20%20%20:%20password/*\ HTTP/1.1
HTTP	214	GET /door/db.php?cmd=Created%20Time%20%20%20%20%20%20:%2010.11.2021%2015:13:38/*\ HTTP/1.1
HTTP	388	GET /d/msdownload/update/software/updt/2021/10/windowspchealthchecksetup_5ede1d4beaf426a8102ef96065bf89758fe40711.cab HTTP/1.1
HTTP	191	GET /door/db.php?cmd=Modified%20Time%20%20%20%20%20:%20/*\ HTTP/1.1
HTTP	275	GET /door/db.php?cmd=Filename%20%20%20%20%20%20%20%20%20%20:%20C:\[redacted]Data\Local\Microsoft\Edge\User%20Data\Default\Login%20Data/*\ HTTP/1.1
HTTP	207	GET /door/db.php?cmd===/*\ HTTP/1.1
HTTP	157	GET /door/db.php?cmd=)*(HTTP/1.1
HTTP	157	GET /door/db.php?cmd=/*\ HTTP/1.1
HTTP	172	GET /wpad.dat HTTP/1.1

Figure 9. Analysis of Network Traffic Received While Malicious Operation

Unique characters from all analyses described here and unique to this application are parsed and noted. These unique string values are then converted to either hexadecimal or used as plain text. In addition to this information, summary function values of the malware are also noted.

4.3. YARA rule for malware analysis

Figure 10 shows the YARA detection rule for the malware analyzed in Section 4.1.

```
rule Dropper_dekont {
    meta:
        description = "dekont - file dekont.exe"
        date = "2021-12-08"
        hash1 = "ca75d22f12250ac4f3b156569655f8a181e9ef10f32befb8a63a0f13773e810d"
    strings:
        $s2 = "/AutoIt3ExecuteScript" fullword wide
        $s4 = "/AutoIt3ExecuteLine" fullword wide
        $s5 = "PROCESSGETSTATS" fullword wide
        $s6 = "WINGETPROCESS" fullword wide
        $s7 = "SCRIPTNAME" fullword wide /* base64 encoded string 'H$H=3@0' */
        $s8 = "SHELLEXECUTE" fullword wide
        $s9 = "SHELLEXECUTEWAIT" fullword wide
        $s10 = "*Unable to get a list of running processes." fullword wide
        $s11 = "HTTPSETUSERAGENT" fullword wide
        $s12 = "PROCESSCLOSE" fullword wide
        $s13 = "PROCESSEXISTS" fullword wide
        $s14 = "PROCESSLIST" fullword wide
        $s15 = "PROCESSSETPRIORITY" fullword wide
        $s16 = "PROCESSWAIT" fullword wide
        $s17 = "PROCESSWAITCLOSE" fullword wide
        $s18 = "PROCESSORARCH" fullword wide
        $s19 = "STRINGREVERSE" fullword wide /* base64 encoded string 'I4H4dDTDR' */
        $s20 = "Error parsing function call.0Incorrect number of parameters in function call.'\"ReDim\" used without an array variable.>Illegal " wide
        $o1 = "IsDebuggerPresent"
        $o2 = "VirtualAlloc"
        $o3 = "InternetOpenUrlW"
        $o4 = "InternetReadFile"
        $o5 = "HttpOpenRequestW"
        $o6 = "HttpSendRequestW"
        $o7 = "FtpOpenFileW"
        $o8 = "LoadLibraryA"
        $o9 = "GetProcAddress"
        $o10 = "ReadFile"
        $o11 = "WriteFile"
        $o12 = "GetTempPathW"
    condition:
        uint16(0) == 0x5a4d and filesize < 5000KB and
        8 of ($s*) and all of ($o*)
}
```

Figure 10. Malware YARA Detection Rule Analyzed in Section 4.1

5. Discussions

In this study, we investigated the attackers' techniques and persistence mechanisms used in cyber attacks on an organization's systems using the YARA model, an open-source tool. YARA is a powerful tool developed by a private company that provides a rule-writing format and has been made available for general use. We contribute to the literature by developing special rules for YARA that are accessible to the general public. Our study enables cybersecurity experts to take preventative measures against potential cyber-attacks in their organizations, helping to detect and respond to such attacks quickly. This study involved the analysis and investigation of Windows and Linux operating systems. Tests were conducted on server and end-user deployments of these systems, where the most common types of attacks were applied. Various malware was installed on the computers using these attack techniques to ensure persistence for continuous unauthorized access to the systems. Detection and removal of malware on the compromised systems were demonstrated using manual techniques with the help of YARA, a detection tool that can develop rules based on its structure. Rules were developed based on the obtained outputs to detect different types of malware using open-source solutions.

6. Conclusions

The present study underscores the significance of comprehending the techniques of attackers and their persistence mechanisms utilized in cyber assaults on an organization's systems. Specifically, we employed YARA, an open-source tool, to create exclusive rules accessible to the general public. Cybersecurity professionals can use these rules to detect and respond to potential cyber-attacks rapidly. The investigation encompassed an analysis of Windows and Linux operating systems, and multiple categories of malware were installed on computers using frequent attack techniques to ensure persistence for sustained unauthorized access to the systems. Our study demonstrates the effectiveness of YARA in detecting and removing malware on compromised systems. In summary, the discoveries of our study can serve as a foundation for organizations to safeguard their systems better and deter cyber attacks.

7. Author Contribution Statement

The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

8. Ethics Committee Approval and Conflict of Interest

"There is no conflict of interest with any person/institution in the prepared article"

9. References

[1] Abomhara M, Køien GM. "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks". *Journal of Cyber Security and Mobility*, 65–88, 2015.
[2] Eggers S. "A novel approach for analyzing the nuclear supply chain cyber-attack surface". *Nuclear Engineering and Technology*, 53(3), 879-887, 2021.
[3] Freilin FC, Holz T Wicherski G. "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks". *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*, 2005: Springer, 319-335.
[4] Auty M. "Anatomy of an advanced persistent threat". *Network Security*, 4, 13-16, 2015.
[5] Ahmad A, Webb J, Desouza KC, Boorman J. "Strategically-motivated advanced persistent threat: Definition, process, tactics and a disinformation model of counterattack". *Computers & Security*, 86, 402-418, 2019.
[6] Schneier B. "The future of incident response". *IEEE Security & Privacy*, 12(5), 96-96, 2014.
[7] Bhatt P, Yano ET, Gustavsson P. "Towards a framework to detect multi-stage advanced persistent threats attacks". in *2014 IEEE 8th international symposium on service oriented system engineering*, IEEE, 390-395, 2014.
[8] Itodo C, Varlioglu S, Elsayed N. "Digital forensics and incident response (DFIR) challenges in IoT platforms". *4th International Conference on Information and Computer Technologies (ICICT)*, IEEE, 199-203, 2021.
[9] Johansen G. *Digital forensics and incident response*. Packt Publishing Ltd, 2017.
[10] Kim S, Kim J, S, Kim D. "WebMon: ML-and YARA-based malicious webpage detection". *Computer Networks*, 137, 119-131, 2018.
[11] Kumar MS, Ben-Othman J, Srinivasagan K. "An investigation on wannacry ransomware and its detection". in *2018 IEEE Symposium on Computers and Communications (ISCC)*, IEEE,1-6, 2018.
[12] Rosyid NR, Murti BB, Prayudha B, Ramadloni AF, Subekti L. "Malware Detection on local network based on honeypot and Yara". *Sistemasi: Jurnal Sistem Informasi*, 12(1), 186-193, 2023.
[13] Siddabathula KS, Panneerselvam RK, Vasana V, Vejendla J, Rafi M, Gummadi SB. "YaraCapper–YARA rule-based automated system to detect and alert network attacks". in *Research Advances in Network Technologies*: CRC Press, 25-47.
[14] Si Q. *et al.*, "Malware detection using automated generation of yara rules on dynamic features". in *Science of Cyber Security: 4th International Conference, SciSec 2022,Matsue, Japan, August 10–12, 2022, Revised Selected Papers*, Springer, 315-330, 2022.

[15] Naik N, Jenkins P, Savage N, Yang L, Naik K, Song J. "Embedding fuzzy rules with YARA rules for performance optimisation of malware analysis". in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 1-7, 2022.
[16] Khalid M, Ismail M, Hussain M, Durad MH. "Automatic yara rule generation". in *2020 International Conference on Cyber Warfare and Security (ICCWS)*, IEEE, 1-5, 2020.
[17] Xu L, Qiao M. "Yara rule enhancement using Bert-based strings language model". in *2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, IEEE, 221-224, 2022.
[18] Naik N. *et al.*, "Embedded YARA rules: strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis". *Complex & Intelligent Systems*, 7, 687-702, 2021.
[19] Naik N, Jenkins P, Cooke R, Gillet J, Jin Y. "Evaluating automatically generated YARA rules and enhancing their effectiveness," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 1146-1153, 2020.
[20] Raff E. *et al.*, "Automatic YARA rule generation using biclustering". in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, 71-82, 2020.
[21] Bilstein D, Plohmann D. "YARA-signator: automated generation of code-based YARA rules". *J. Cybercrime Digit. Invest.*, 5(1), 1-13, 2019.
[22] Nguyen NH, Le VH, Phung VO, Du PH. "Toward a deep learning approach for detecting php webshell". in *Proceedings of the 10th International Symposium on Information and Communication Technology*, 514-521, 2019.
[23] Yusof AR, Udzir NI, Selamat A. "Systematic literature review and taxonomy for DDoS attack detection and prediction". *International Journal of Digital Enterprise Technology*, 1(3), 292-315, 2019.
[24] Yin D, Zhang L, Yang K. "A DDoS attack detection and mitigation with software-defined Internet of Things framework". *IEEE Access*, 6, 24694-24705, 2018.
[25] Joshi B, Vijayan AS, Joshi BK. "Securing cloud computing environment against DDoS attacks". in *2012 International Conference on Computer Communication and Informatics*, IEEE, 1-5, 2012.
[26] Chiew KL, Yong KSC, Tan CL. "A survey of phishing attacks: Their types, vectors and technical approaches". *Expert Systems with Applications*, 106, 1-20, 2018.
[27] Tandale KD, Pawar SN. "Different types of phishing attacks and detection techniques: A review". in *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, IEEE, 295-299, 2020.
[28] Le Page S, Jourdan GV, Bochmann GV, J Flood, Onut IV. "Using url shorteners to compare phishing and malware attacks". in *2018 APWG Symposium on Electronic Crime Research (eCrime)*, IEEE, 1-13, 2018.
[29] Pirscoveanu RS, Hansen SS, Larsen TM, Stevanovic M, Pedersen JM, Czech A. "Analysis of malware behavior: Type classification using machine learning." in *2015 International conference on cyber situational awareness, data analytics and assessment (CyberSA)*, IEEE, 1-7, 2015.
[30] Blaise A, Bouet M, Conan V, Secci S. "Detection of zero-day attacks: An unsupervised port-based approach". *Computer Networks*, 180, 107391, 2020.
[31] Kim JY, Bu SJ, Cho SB. "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders". *Information Sciences*, 460, 83-102, 2018.
[32] Conti M, Dragoni N, Lesyk V. "A survey of man in the middle attacks". *IEEE communications surveys & tutorials*, 18(3), 2027-2051, 2016.
[33] Tommasi F, Catalano C, Taurino I. "Browser-in-the-Middle (BitM) attack". *International Journal of Information Security*, 21(2), 179-189, 2022.
[34] Alberto O, Marco V. "Man in the middle attacks". in *Blackhat Conference Europe*, 2003.
[35] Lee J, Lee S. "A study on unknown malware detection using digital forensic techniques". *Journal of The Korea Institute of Information Security & Cryptology*, 24(1), 107-122, 2014.
[36] Bazrafshan Z, Hashemi H, Fard SMH, Hamzeh A. "A survey on heuristic malware detection techniques". in *The 5th Conference on Information and Knowledge Technology*, IEEE, pp. 113-120, 2015.
[37] Duby A, Taylor T, Zhuang Y. "Malware family classification via residual prefetch artifacts" .in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 256-259, 2022.
[38] Naik N. *et al.*, "Fuzzy hashing aided enhanced YARA rules for malware triaging". in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 1138-1145, 2020.
[39] Culling C. "Which YARA rules rule: basic or advanced?". *GIAC (GCIA) Gold Certification and RES*, 5500, 2018.