applied sciences

Review

Classification and Analysis of Malicious Code Detection Techniques Based on the APT Attack

Kyungroul Lee [1], Jaehyuk Lee [2] and Kangbin Yim [3,*]

1 Department of Information Security, Mokpo National University, Mokpo 58554, Republic of Korea
2 Interdisciplinary Program of Information & Protection, Mokpo National University, Mokpo 58554, Republic of Korea
3 Department of Information Security Engineering, Soonchunhyang University, Asan 31538, Republic of Korea
* Correspondence: yim@sch.ac.kr; Tel.: +82-41-530-1741

Abstract: According to the Fire-eye's M-Trends Annual Threat Report 2022, there are many advanced persistent threat (APT) attacks that are currently in use, and such continuous and specialized APT attacks cause serious damages attacks. As APT attacks continue to be active, there is a need for countermeasures to detect new and existing malicious codes. An APT attack is a type of intelligent attack that analyzes the target and exploits its vulnerabilities. It attempts to achieve a specific purpose, and is persistent in continuously attacking and threatening the system. With this background, this paper analyzes attack scenarios based on attack cases by malicious code, and surveys and analyzes attack techniques used in attack cases. Based on the results of the analysis, we classify and analyze malicious code detection techniques into security management systems, pattern-based detection, heuristic-based detection, reputation-based detection, behavior-based detection, virtualization-based detection, anomaly detection, data analysis-based detection (big data-based, machine learning-based), and others. This paper is expected to serve as a useful reference for detecting and preventing malicious codes. Specifically, this article is a surveyed review article.

Keywords: malicious code; detection technique; attack scenario; attack technique; APT attack

Citation: Lee, K.; Lee, J.; Yim, K. Classification and Analysis of Malicious Code Detection Techniques Based on the APT Attack. *Appl. Sci.* **2023**, *13*, 2894. https://doi.org/10.3390/app13052894

Academic Editors: Christina Thorpe and Stephen O' Shaughnessy

Received: 11 January 2023
Revised: 21 February 2023
Accepted: 22 February 2023
Published: 23 February 2023

1. Introduction

The APT (Advanced Persistent Threat) attack refers to a specific designation of security threat created by the U.S. Air Force Command in 2006 to facilitate smooth communication with government agencies. It is a type of global hacking attack [1,2] in which a specific company or organization is targeted; it is a threat to certain protected targets that takes the form of a stealthy and continuous attack. In contrast to conventional cyber hacking attacks, an APT attack is characterized by an intelligent and continuous threat. The National Institute of Standards and Technology (NIST) has defined an APT attack as "a cyberattack that uses a considerable amount of expertise and resources, and creates opportunities with various attack methods to achieve specific goals [3]". Unlike traditional cyber threats, which can be remotely controlled and executed, such as bots or system malware that are intended to destroy systems and leak information, an APT attack takes a long time to detect, because the target is clear and is continuously attacked at various times [4,5]. In particular, the IoT or Unmanned Aerial Vehicle (UAV) environment, which is a small environment that uses embedded systems, is not safe from security threats launched by attackers, and there is a need for methods that can be used to detect such attacks.

In this type of attack, the attacker finds weaknesses through social engineering techniques, which they use to construct elaborate plans to fulfil the final purpose of the APT attack, and then stealthily penetrate the system based on the identified vulnerabilities. To increase its success rate, this type of attack explores the target system and constantly attempts to infect it with malicious codes through zero-day vulnerabilities and rootkit techniques. It also combines attack techniques such as DBD (Drive-By-Download), SQL injection, malware and spyware, phishing, spam, spear phishing, watering holes, and manual attacks, if the victim does not recognize the attacks [5–7]. An APT attack is defined as an intelligent (Advanced) analysis of vulnerabilities that satisfies the conditions of Persistent attacks, which involve multiple continuous attacks to achieve a specific purpose and to Threaten protected targets [4,5,8].

The Fire-eye's M-Trends Annual Threat Report 2022 classifies the factors of a recent APT attack as outside and inside, and these are listed in Table 1.

Table 1. Ransomware Threat Report.

Infection Notice	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Outside	94%	63%	67%	69%	53%	47%	38%	41%	53%	41%	47%
Inside	6%	37%	33%	31%	47%	53%	62%	59%	47%	59%	53%

The table above presents the APT attack detection ratio for solutions installed by APT defense solution vendors. Here, 'outside' of the infection notification indicates that the APT attack has been notified from the outside, while 'inside' of the infection notification means that the APT attack has been detected from the inside. Examining this trend, 94% of APT attacks were notified from the outside in 2011, but this value significantly decreased to 53% in 2019. In other words, although the rate of attacks being detected internally has increased through defense technology and the introduction of solutions against APT attacks, 50% of APT attacks are still notified of infection from the outside, which is still substantial [9].

Therefore, we analyze attack scenarios based on the above cases of malicious codes and survey, and we analyze the specific techniques used in these attacks. Based on the results of this analysis, we classify malicious code detection methods into security management systems, pattern-based detection methods, heuristic-based detection methods, reputation-based detection methods, behavior-based detection methods, virtualization-based detection methods, anomaly detection methods, big data-based detection methods, and other methods. The rest of this paper is organized as follows: The "Attack cases by malicious codes" section describes attack cases caused by existing malicious codes such as APT, and the "Malicious code attack scenario" section describes attack scenarios that involve penetrating into the system with malicious codes. The "Attack techniques used by malicious code" section explains the attack techniques that are employed when using malicious codes. Finally, the "Classification and analysis of malicious code detection methods" section describes the classification and analysis of the methods used to detect and thwart malicious codes.

The contributions of this study are as follows:

- This paper presents attack scenarios according to malicious codes based on APT attack cases. Moreover, according to the results of the scenarios, methods to detect malicious codes were classified and analyzed into security management systems, pattern-based detection methods, heuristic-based detection methods, reputation-based detection methods, behavior-based detection methods, virtualization-based detection methods, abnormal detection methods, big data-based detection methods, and other methods.
- Moreover, this article is distinct from survey articles comparing existing malicious code detection methods as follows: 1. We focus on the latest malware detection technologies. 2. We classify and analyze overall malicious code detection technologies, without being limited to a specific method. 3. Scenarios were presented based on actual attack cases, and malware detection methods were classified based on the presented scenarios.
- Finally, through the classification and analysis results of malicious code detection methods, we expect this article to be helpful in developing effective malicious code detection technologies against various attacks in the future.

2. Attack Cases by Malicious Codes

Known APT attack cases rely on using several types of malicious code. In South Korea, there have been many major hacking attempts, such as the IceFog, H company, N company, S company, 3.20 cyber terror, and 6.25 hacking incidents [10–16]. IceFog was an APT attack that was detected in 2011 and which targeted research institutes, the defense industry, and mobile service companies, from which it stole information. This information included basic system information such as folder lists, adapter lists, IP configurations, and network information, along with sensitive information such as Windows address book (WAB) files, documents, user account credentials, etc. Spear phishing emails were used, which induced recipients to connect to malicious codes or malicious websites by clicking the links in the emails. Then, the malicious code sent in the emails infected the target systems by exploiting vulnerabilities in programs such as Microsoft Office. After infecting a system, the characteristics of the system were examined and investigated. If the system was a virtual machine or a fake system, the malicious code did not perform any malicious behaviors. However, if it was an actual system, the target information was captured [10].

In April 2011, N company suffered an APT attack, which resulted in the deletion of a large amount of data and paralysis of the target network. The attacker inserted malicious code into the target system by distributing free download coupons from websites engaged in malicious behaviors. After inserting 81 additional malicious codes into the laptop of the subcontractor who maintained the server, the attacker stole sensitive information that could be used to access the internal network, such as the administrator password. Malicious codes for eavesdropping and receiving attack commands via remote control were also installed in the same manner, and the server inside the N company was destroyed by the installed malicious code after receiving commands transmitted from the outside. The attacker deleted files with 31 specific extensions, then destroyed the hard disk. Many hard disks were attacked, and ultimately, 275 internal servers, 45 of 98 web servers, 180 internal servers, and 48 of 49 test servers were destroyed in this attack [11,12].

A malicious attack in July 2011 resulted in the leakage of the personal data of 35 million individuals from S company. The leaked information included the ID, encrypted password, personal number, name, date of birth, sex, e-mail address, telephone number, and address of the individuals. The attacker hacked a public software update server to install malicious code, then collected internal information related to the database manager of any insider who downloaded this malicious code. After that, the attacker used this information to steal data stored in the database from a remote location [11–13].

The 3.20 cyber terror attack in March 2013 caused paralysis of the target network by erasing data from major broadcasters and financial institutions. In this attack, the attacker hacked the web server and infected it with malicious code. Next, the malicious code was distributed to the PCs connected to this web server. Then, the attacker stole the administrator account of the antivirus software distribution server. In total, 89 malicious codes were placed on PCs, antivirus software distribution servers, and ATMs, and the hard disks of these machines were destroyed after files in a specific folder were deleted. The attacker damaged 48,862 hard disks in total, thus resulting in network paralysis. More importantly, anti-virus software and security solutions were running on these systems, and measures such as deployment of United Threat Management (UTM) have been taken to prevent intrusions from the outside. Here, UTM means a comprehensive management tool that includes various security tools such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), firewall, etc. [17]. Nevertheless, these solutions did not effectively defend against this attack. This is because these types of defense systems can effectively prevent intrusion by known malicious codes, but unknown attacks are difficult to defend against. The vulnerabilities that were exploited in the 3.20 cyber terror attack include the acquisition of administrator authentication authority, the absence of non-repudiation and integrity check in distribution of general software files, and the lack of a protection function in the agent itself. These vulnerabilities prevented the detection and treatment of malicious code. Therefore, the attacker was able to remotely connect to the system, which allowed for remote control and communication of the infected system. Then, after additional malicious code was injected through the dropper and downloader function, the system and MBR were destroyed [11–13]. In addition, the private information of about 1.75 million customers was leaked from H company in April 2011 in the 6.25 hacking incident, and there were also attacks on the opening ceremony of the 2018 PyeongChang Olympics [11,13–16,18].

International attack cases include Red October, eBay, Night Dragon, GhostNet, Operation Aurora, Duku, Stuxnet, RSA, Flame, TARGET, etc. [10–13]. Red October is an APT attack that has been active since 2007; in this attack, malicious code has attacked governmental, research, energy, and nuclear organizations in several countries. The attacker sent an e-mail containing malicious code that exploited vulnerabilities in Microsoft Office and Excel to penetrate the target system, and the malicious code infected the victim's system. The infected system leaked sensitive information such as stored information, deleted information, and network equipment setting information that was stolen through modules. It leaked data from mobile devices such as smart phones as well as PCs [10].

Night Dragon was an APT attack that occurred from 2009 to 2011. In Night Dragon, Chinese servers leaked information about gas and oil production systems, oil exploration documents, and industrial control systems in Kazakhstan, Taiwan, Greece, and the United States. The attacker used social engineering techniques, Windows vulnerabilities, the Active Directory, and Remote Administration Tools (RATs), and they attempted to infect malicious code after uploading that malicious code through an SQL injection attack on a web server. After obtaining the account information to access the system, the attacker tried to access and collect sensitive information such as the account information of users in the system [10]. Another attack, the Operation Aurora APT attack, was discovered in January 2010. It took over software source code and critical data for telecommunications companies. In addition to social engineering techniques, the attacker used spy phishing to link to a malicious Web site using MS-10-002, a zero-day vulnerability in Internet Explorer. When the victim accessed the web site, the attacker gained access to the internal system through malicious JavaScript files, through which they could steal the source code and important data [10,12]. The Stuxnet APT attack was discovered in June 2010. It aimed to destroy the system of a nuclear power plant. The attacker remotely manipulated the remote monitoring and control system in the power plant. To this end, the attacker exploited Windows shell LNK, Windows server service, Windows print spooler, and shared network service vulnerabilities. Moreover, this attack used a rootkit to hide itself and propagated malicious code in a self-replicating manner. In a more concerning APT attack, a USB storage device was used as an attack tool to physically penetrate into a separate network [10,11,13]. Duku, another APT attack that was similar to Stuxnet, was discovered in September 2011, but its purpose was to spy on users instead of destroying the system. The attacker collected authentication information such as passwords by key logging in the penetrated system, and they spied on the target by additionally accessing other systems by using the stolen information. This attack was difficult to detect because it penetrated the system using a legally signed certificate [10]. Flame was the first APT attack discovered in 2012; it took information from the Windows system used by the target organization. The attacker stole confidential information such as keyboard inputs, information displayed on the screen through screen capture, e-mails, Bluetooth device information, and conversation information recorded by a microphone [10]. There have been many additional attacks that have stolen important information, such as TARGET [11], the large-scale cyber-spying case GhostNet [13], RAN hacking [11], and eBay hacking [13]. WannaCry, one of the most known ransomware attacks, is a representative APT attack that was revealed in May 2017. It was distributed through large-scale cyber-attacks, ultimately infecting about 120,000 computers in about 100 countries around the world. The attacker produced ransomware using an NSA program through social engineering techniques and caused massive damage worldwide by exploiting the vulnerability of Eternal Blue, Microsoft's Windows operating system's SMB vulnerability [19–21].

3. Malicious Code Attack Scenario

APT attack scenarios can be classified into six stages: preliminary investigation, penetration, latency, internal search, internal activity, and purpose achievement. This attack scenario is illustrated in Figure 1.

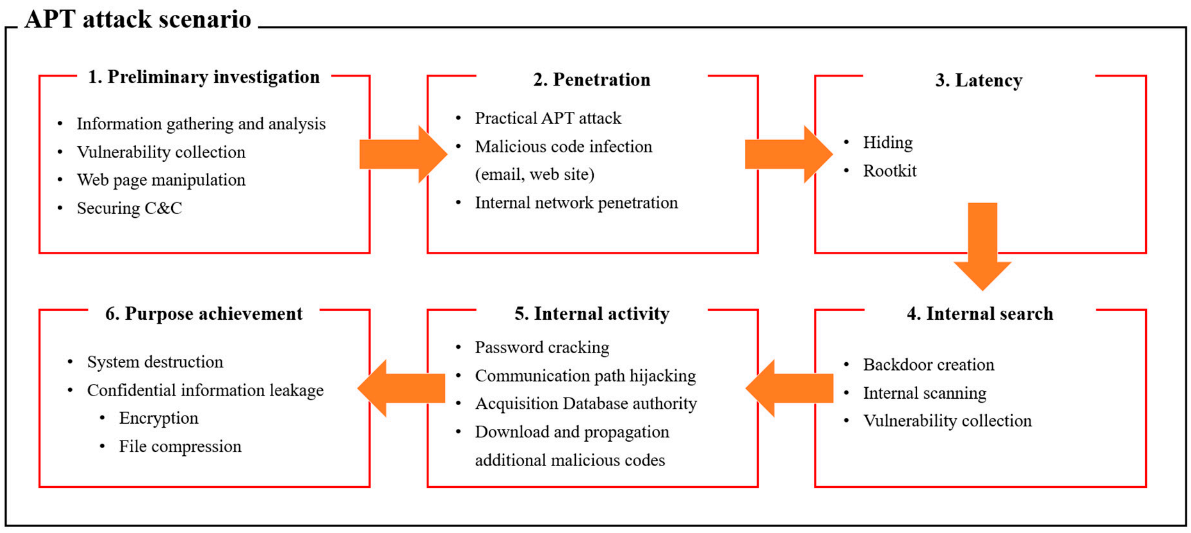

Figure 1. APT attack scenario.

In the preliminary investigation step, the attacker prepares for the future attack phases; this may include information gathering and analysis in addition to other types of attack preparation. The information gathering and analysis process involves collecting information such as the homepage of the attack target, Social Networking Service (SNS) [12], internal organizational structure, employee name and contact information, type of software, partner companies, frequently accessed web sites and Storage as a Service (StaaS), etc. Using the collected information, it is necessary to use an attack preparation process to collect vulnerability information for future penetration, and the attacker acquires a base for attack through this process. In the attack preparation process, to collect vulnerability information, the attacker obtains information related to the people who have access to the secure information necessary for the attack by investigating intrusive ports using technical port scan or non-technical social engineering techniques. Once vulnerability information is gathered and the target information is determined, the attacker creates a dedicated injection tool and writes malicious code that exploits the vulnerabilities of the target. The attacker also manipulates frequently accessed web sites and web hardware to upload the attack tool and malicious code. Then, when the attacked internal employee accesses the manipulated web sites, the uploaded malicious code is downloaded and successfully penetrates the system. Moreover, an external attacker can procure a C & C (Command & Control) server for communication with the internal network, and they can prepare for attacks through spear phishing and USB infection for future penetration. The preliminary investigation step is not itself an attack, because it mostly involves information gathering. It is therefore difficult to effectively detect attacks at this stage [10–13,22].

In the penetration step, the attacker infects the victim's system with malicious code generated by the attacker based on the information collected in the previous preliminary investigation step. Next, the internal network is penetrated into, thereby practically performing the attack, and the target system is infected with malicious code, mainly by using social engineering techniques. The attacker mainly targets the administrator of the internal organization, the main system, the security officer, and the network manager based on the information collected in the previous step, as these people have the authority to access security information. To infect the systems of these victims, the preparation for the penetration in the previous step is completed by uploading specially designed malicious code, such as is the case in a zero-day attack, which involves bypassing anti-virus software, collecting information, and communicating with the C&C server. This step infects malicious code into the internal system through a spear phishing attack that send e-mails to the selected victims, and by opening up access to manipulated web sites. The attacker sends items such as personal information, social issues, personal card bills, and shopping mall delivery information to the e-mail to get the victims to read the e-mail and/or execute the attachment file. In this way, the target system is infected with malicious code through the drive-by download technique and watering hole attachment technique by executing the attached malicious code or inducing a connection to the manipulated web site and update server. Therefore, a defender is limited in their ability to detect such an attack in this step because the attacker uses social engineering techniques and zero-day vulnerabilities. In fact, it was possible to penetrate the internal network by neutralizing the internal security system in the case of the S company and 3.20 cyber terror attacks, which are typical APT attack cases in South Korea [10–13,22–24].

In the latency step, the malicious code that penetrated the system in the penetration step is hidden from detection. Because of this, the malicious code is not detected by the security system, so continuous and additional penetration is possible. To this end, the rootkit technique, which is a general technique that malware uses to conceal itself, is utilized. It is also possible to pretend to be a normal user by acting on legitimate accounts, protocols, and time zones [10,11,13,25].

In the internal search step, the attacker prepares for the next attack step by collecting information inside the system, such as information on the internal system and infrastructure. A backdoor for communication with an external attacker is also installed in this step. Using this backdoor, the attacker infects the system with additional malicious codes and collects information from the system and network through scanning. The attacker then prepares for internal activities by identifying vulnerabilities based on this information. Regarding the malicious code penetration in the previous step, the intent of penetration is often strong and does not include a function for internal search, such that the attack is not detected by the security system. Therefore, the attacker installs a backdoor into the infected system for internal search, and the malware attempts to communicate with the external attacker through this backdoor. In this process, the backdoor utilizes technologies such as cryptography to avoid being detected by the security system, as the sent data is encrypted. Upon completion of the backdoor installation, communication with the attacker becomes possible. Therefore, the preparation for the internal search is completed, and a scanning tool and vulnerability analysis tool are installed for this purpose. Scanning tools are tools that collect information about internal systems and infrastructures. These tools can collect confidential information through legal commands from the internal system so that they are not detected by the security system. From the attacker's perspective, it is possible to collect internal information in a stealthier manner, because this attack attempts to acquire information from previously infected systems. It also collects information about the systems that are authorized to prepare for future attacks, and it analyzes the vulnerabilities to take control of the target system and other systems. Through these processes, it is possible to propagate malicious code to vulnerable and privileged systems, and the attacker continues to perform the above steps to achieve their final goal [10,11,13,22–24].

The internal activity step is the final preparation step. In this step, the attacker penetrates the target system through acquisition of authority, propagation of malicious code, and the downloading of additional malicious code. In the previous and current steps, all of the information and vulnerabilities are collected based on the privileges of the malicious code installed by the attacker, and the privilege to penetrate the target system is obtained through these acts. Spy phishing, password cracking, and communication path hijacking attacks are performed by internal employees to gain authority [13]. In these processes, additional malicious code is required for each attack, so various malicious codes for each action are additionally downloaded, and malicious codes are propagated to vulnerable internal systems and networks to gain additional authority. In this way, it is possible for malicious code to infect partial systems or entire systems within the network. Therefore, the attacker prepares for the final attack by obtaining the authority of the database and the target system. Moreover, in this process, the infected system is continuously monitored and manipulated to avoid detection by the security system [10,11,13,21,22,24].

In the purpose achievement step, the attacker actually achieves their desired goal, mainly by destroying the system or leaking confidential information. In other words, the attacker achieves the final goal by sending specific commands or setting a time for execution. For example, the attacker can bypass the defense system using techniques such as encryption, file compression, and file partitioning to disguise their activities as normal traffic. There is the potential for detection by the defense system during information transmission in the case of information leakage. In the case of system destruction, the purpose is achieved by interfering with the normal use of the infected system or destroying the equipment by manipulating the file system or deleting the hard disk [10–13,21–25].

4. Attack Techniques Used by Malicious Code

APT attack techniques can be categorized into those pertinent to the preliminary investigation step, penetration step, latency step, internal search step, and internal activity step. The following Table 2 lists the techniques that are used at each stage.

Table 2. Attack techniques using malicious codes.

Attack Step	Attack Techniques
Preliminary investigation	Preparation of manipulated websites SQL injection Web vulnerability Malware and Spyware Malware and Spyware Zero-day vulnerability
Penetration	Mail (phishing, spam, spear phishing) DBD (Drive-By-Download) Watering hole Manual attack
Latency	Mail (phishing, spam, spear phishing) DBD (Drive-By-Download) Watering hole
Internal search	Backdoor Network monitoring and analysis System monitoring
Internal activity	Keylogger IRC Interception of password hash values of victim accounts Eavesdropping on conversations recorded by microphone Information exposure and leakage

The preliminary step involves preparing for penetration. Commonly, an attacker makes a manipulated web site and prepares it for penetration. To manipulate a web site, the attacker uses attack techniques that exploit SQL injection and Web vulnerabilities. SQL injection is a code injection attack technique that manipulates a database by causing a malicious SQL query to be executed. Some attackers have attempted to change the target's homepage screen [11–13,26]. An example of an attack that exploits Web vulnerabilities is one wherein an environment is built in which malicious code is penetrated by vulnerable Web sites that are frequently visited by victims. When accessing the manipulated website, a malicious code is downloaded to the victim's system [13]. To achieve such an attack, malicious codes must be inserted into the prepared website; these codes include malware [11,27], spyware [11,26], and code that exploits zero-day vulnerabilities [10,13,26,28]. In the latter, the attacker attempts to exploit a zero-day vulnerability that has been discovered in systems, protocols, operating systems, and applications, but has not been addressed. It does not take much time to write new malicious code that utilizes zero-day vulnerabilities, so target systems often cannot defend against such an attack because the new malicious code cannot be detected by anti-virus systems that look for existing malware signatures. In fact, at the end of March 2014, Microsoft published a Security Advisory (MS Security Advisory 2953095) detailing vulnerabilities in RTF files in MS-WORD 2010, and also provided examples of RTF files that utilized this vulnerability and were attached to many spam emails [10,13,26,28].

The penetration step is a practical step in which an attacker penetrates into the target system by downloading from a website or malicious code prepared in advance. Attackers may attempt to attack via email, such as in phishing, spam, spear phishing, drive-by-download, watering hole, and manual attacks. Penetration by email is an attack technique that consists of inviting users to connect to a malicious server and executing a file when they click a malicious web site address or path, or when they click a malicious file attachment; this type of attack is mainly done through phishing or spam emails [11,13,26,29,30]. However, attackers have developed more reliable attack techniques since some phishing and spam emails can now be detected by filters. For example, spear phishing has emerged. Spear phishing, unlike traditional phishing and spam email, involves sending emails to a victim that appear to have originated from a person associated with the victim, or another trusted source such as the government or a business. The attack proceeds when the victim opens an attachment or clicks on a corrupted website address. Spear phishing is a serious problem because the malicious emails are made to resemble normal emails, thus causing unsuspecting users to click on links and files [10,11,13]. Drive-by download is a method of attacking a vulnerable web site and infecting through simple access of the web site. It is characterized by the difficulty of recognizing lard spreads and infections, and it is one of the most threatening attack techniques. In this technique, malicious code is inserted into the website using an iframe and JavaScript, and it occasionally includes a waypoint to hide the final distribution site. The attacker infects the target system through the downloading and execution of malicious code that includes obfuscated JavaScript files, which are difficult to detect. An attack using iframe attempts to attach an iframe to the main html file, then proceeds as a drive-by download attack; the iframe is hidden by setting the width and height to zero so the user does not see it [11–13,26,28,29]. A watering hole is a type of drive-by-download attack that is characterized by attacking a specific target. This attack is very dangerous because it attempts to exploit unknown vulnerabilities in general [13,29]. As described above, the attacker penetrates a victim's system in an automated manner, but they sometimes perform a more sophisticated manual attack. The attacker operates on the basis of information collected in the preliminary investigation step, and they write and use an automated tool or specific commands to propagate the malicious code [13]. There are also IP clocking attacks [12], Layer-1 attacks [12], etc.

The infection paths in the penetration step include malicious code infection through visits to internal websites, malicious code infection through regular website visits, email attacks with specific goals inside the target organization, malicious code infection through the media, and management system attacks. Through these paths, the malicious code prepared in the above process is installed on and penetrates the victim's system [13].

In the latency step, the malicious code that is placed on the victim's system in the penetration step is hidden to avoid being detected by the security system. In this step, the malicious code makes use of attack techniques such as rootkit, encryption, memory patch, steganography, code obfuscation, execution compression, and anti-VM (Virtual Machine) [13,26,28]. A rootkit hides malicious code so that it cannot be detected by the security system; it mainly tries to block and conceal access by manipulating the legal operations in the user and kernel modes. For example, the malicious code hooks API functions such as IoCompleteRequest() to block access to programs and files, and it hides its own process by changing the process structure (EPROCESS) [13,26,28]. Encryption is a technique in which an attacker encrypts a program that has successfully penetrated the target system so that it is not detected by the security system, or encrypts the contents to hide packets that are transmitted to the network. Since the encrypted program cannot execute malicious behaviors until the program has been decrypted, it can remain hidden and go undetected by the security program. The encrypted packets can pass through the security system and be securely transmitted to the victim's system because they cannot be checked until the packets are decrypted. Moreover, malicious actions can be performed, such as specific attacks and information gathering through communication with the outside [13,28]. A memory patch is a technique for inserting hidden code in an area that loads into the memory in order to conceal a program that an attacker has successfully installed on the target system. The memory patch mainly relies on code injection and DLL injection techniques. This technique does not exist as a real file, but because the code is loaded into memory, defenders directly explore and detect memory areas. Because of this, there are cases in which detection is difficult or impossible. For example, code-concealment techniques such as memory patches were used in Regin, a malicious code with the aim of leaking information [28]. Other concealment techniques include steganography [10] to hide communication, code obfuscation to hide execution code [13], execution compression techniques such as packing to prevent analysis, and anti-debugging [28]. Further, some techniques only operate when certain conditions are met [28].

In the internal search step, the internal system and the network that have been penetrated for the final purpose are searched, and techniques such as backdoor, network monitoring and analysis, and system monitoring are utilized. The backdoor allows the attacker to communicate with the victim's system from the outside. It is mainly used to install additional malicious codes or for the purposes of commands delivery and information leakage [13]. The network monitoring and analysis step involves the gathering and analysis of information on the network to penetrate into the victim's system; this is mainly performed using the port scan technique. In a port scan, information about the ports of the internal network is collected. If successful, the port scan bypasses security functions such as access control because it performs scanning on the authorized system. Therefore, this attack technique is very effective because it is not detected by the security system and is generally allowed to communicate internally [12,13,29]. To perform such an internal search, the above malicious codes are additionally installed, thereby collecting system and user information. The collected information includes system connection information, service usage information, pattern information, etc. Using the collected information, the attacker can approach and penetrate the victim's system, then proceed to the internal activity stage [13].

In the internal activity step, the attacker becomes active in the internal system and network to penetrate the final target system. This is carried out using attack techniques including keylogger, IRC (Internet Relay Chat), information exposure and leakage, interception of password hash values of victim accounts, and eavesdropping on conversations through the microphone. A Keylogger records information that is input using a keyboard. Here, the attacker achieves their goal by taking information input from the keyboard or information stored in the database [13]. This is because access to systems and databases is mostly based on ID-password pairs. IRC [16] is a protocol for real-time chat that also provides functions such as conversation and file transfer. However, these functions can be exploited by attackers, who can then execute malicious actions such as information retrieval by sending specific commands or downloading additional malicious codes based on the data collected through internal search [29]. The method used for information exposure and leakage is similar to that used to collect information about internal systems and networks in the internal search step. The difference from the previous step is that more information is gathered and used to penetrate the victim's system, and the attacker achieves their goal by stealing important information such as authorized certificates, system account information and configuration information [29]. The attackers can also achieve their goal by collecting information to penetrate the final target system, such as intercepting the password hash value of victim accounts and eavesdropping on conversations recording using microphones, as in the case of EMC/RSA [13].

5. Classification and Analysis of Malicious Code Detection Methods

The available methods for detecting malicious code can be largely categorized into security control systems, pattern-based detection methods, heuristic-based detection methods, reputation-based detection methods, behavior-based detection methods, virtualization-based detection methods, anomalous symptom detection methods, and data analysis-based detection methods. Table 3 provides detailed information on the techniques that are utilized by each of these detection methods.

Table 3. Classification of malicious code detection methods.

Detection Methods	Detection Techniques	
Security control system	Firewall and web firewall	
	Anti-virus	
	Intrusion detection system	
	Intrusion prevention system	
	Network forensic	
	APT dedicated solution	Network-based
		Host-based
	Log analysis system	EMS
		SIEM
	Honeypot (client honeypot)	Low interaction client honeypot
		High interaction client honeypot
	Prevention of internal information leakage	
	Secure USB	
	Spam mail protection system	
Pattern-based detection	Signature-based	String signature
		Instruction frequency signature
		CFG signature
		Byte sequence signature
	Misuse detection	Signature analysis
		Expert system
		State transition analysis
		Petri Net
	Malicious URL/IP detection	
Heuristic-based detection	Static heuristic	
	Dynamic heuristic	
	Negative heuristic	
	Genetic algorithm	Wildcard comparison
		Mismatching
Reputation-based detection	Reputation-based detection	
	Behavior-based detection	

Table 3. *Cont.*

Detection Methods	Detection Techniques		
Behavior-based detection	Static analysis		
	Dynamic analysis		
	Hybrid analysis		
	System-based		
	Network-based		
	Integrity check technique		
	Behavior monitoring and blocking	Spear phishing email surveillance	
		Frequency of occurrence of opcode sequence	
		Common behavior graph exhibiting malicious code execution behavior	
		Framework based on machine learning methods	
		Association with registry and process information	
	API call behavior detection		
	Entropy-based detection		
	Visualization	Tree map-based	
		Grayscale binary content	
		Function call graph	
		Control flow graph	
		Behavior dependent graph	
	Traffic characteristic		
	Anormal behavior detection		
	System call sequence state-based detection		
	Fuzzy logic and genetic algorithm		
	Emulator		
Virtualization-based detection	Sandbox		
	Profiling		
Anomalous symptom detection	Data mining		
	Statistical method		
	Real-time monitoring		
Data analysis-based detection	Big data-based detection	Intelligence on threats	
		Behavior profiling	
		Data and user monitoring	
		Application monitoring	
		Analysis and interpretation	
	Machine learning-based detection	Supervised learning	Artificial neural network (ANN)
			Deep neural network (DNN)
			Convolutional neural network (CNN)
			Support vector machine (SVM)
		Unsupervised learning	K-means clustering algorithm

Table 3. *Cont.*

Detection Methods	Detection Techniques		
		Reinforcement learning	Q-learning
			Deep Q-Network
			Generative adversarial network (GAN)
Other detection methods	Packer detection and unpacking		
	Malware-clustering technique		

5.1. Detection Techniques in Security Control Systems

Security control systems can be categorized into firewall and web firewall, anti-virus, intrusion detection system, intrusion prevention system, network forensic, APT dedicated solution, log analysis system, honeypot, prevention of internal information leakage, secure USB, and spam mail protection system.

The firewall technique controls malicious traffic by determining whether to allow or block traffic input and output to and from the network based on predefined policies and rules. The available traffic permission and block methods include the black list-based technique that defines policies and rules for malicious traffic as well as the white list-based technique that defines policies and rules for normal traffic. The black list-based technique blocks traffic that has occurred in the past or abnormal addresses and ports, and the white list-based technique allows only traffic that is input and output from the normal addresses and ports. Therefore, with properly defined policies and rules, this technique has the advantage of blocking attacks from inside and outside. However, it does not prevent attacks that do not have defined rules. If the wrong rules are set, then normal traffic will be blocked, and if many rules are defined to prevent various attacks, then the traffic processing performance will be degraded and loss will occur. Although the web firewall technique is similar to the firewall technique, it differs in that traffic is targeted to a web service, HTTP, HTTPS, FTP, etc., and the protection target is only web servers [11,13].

Anti-virus software is software that protects systems from malware. The software analyzes malicious codes and defines characteristic parts of the codes, for example, a hash value of a specific string or file called a signature. If a particular string is found in the malicious code or has the same hash value, then that code is identified as malicious and its execution is blocked. Malicious programs are detected through comparisons of strings or hash values, which is advantageous in that inspection time is past. Nevertheless, there is a possibility of both false positive and false negative results. For example, in some cases, the same string and the same hash value are defined in a normal program, and it takes a lot of time to update the signature. In a case where an unknown new malicious code appears, there is no signature. Moreover, a variant in which the malicious code is changed will not be detected because the string or hash value has changed. Therefore, to overcome the problems of signature-based detection methods, software is currently being developed using heuristic-based detection methods and behavior-based detection methods [10,13,31,32].

An IDS (Intrusion Detection System) detects intrusions in real time. This technique involves analyzing the principle and type of attacks and generating signatures, which is a characteristic for detecting the signature [33–35]. If the same signature is detected, then the attack is identified as an intrusion. However, this system involves certain disadvantages. For example, the system cannot obtain signatures for unknown attacks, multi-level packing and encryption make it difficult to detect signatures, it is difficult to detect obfuscated malicious code, and there is a need for periodic detection pattern management. An IDS may be classified as a network-based system or a host-based system depending on which system the detection is performed. Network-based systems detect signatures based on network traffic, and they provide network traffic monitoring and usage information capabilities, multi-level attack investigation capabilities, management capabilities for rapid response to external attacks, availability and clustering capabilities, and intrusion detection capabilities. Host-based systems detect intrusions by detecting the signature of the file on the host or analyzing the behavior information of the running system. They also perform resource information file analysis, accurate detection and analysis, real-time log analysis, and the detection of attack attempts at the internal user and user levels. Misuse detection and abnormal detection are classified in terms of the analytical methods they involve. Misuse detection is a method of detecting malicious code by defining rules of known behavior and signatures, such as in signature-based detection techniques. Abnormal behavior detection is a method of detecting malicious behavior when a behavior is not included in a benign model after defining a model of benign behavior in a network and host. Data mining and statistical methods can be used to define models of benign behavior [11,13,22,29].

An IPS (Intrusion Prevention System) is very similar to an intrusion detection system. However, unlike IDS, this technique blocks IP addresses and ports to prevent intrusion as much as possible. Therefore, the advantages and disadvantages of IDS are directly inherited by IPS. Currently, multi-segment technology is widely used to divide one piece of equipment into several pieces of equipment, and various techniques to prevent intrusion are actively being developed [11].

The purpose of network forensic techniques is not to detect and prevent intrusions, but to provide additional information for the forensic analysis of intrusions. This technique collects and stores all traffic input and output data from and to the network, analyzes the stored traffic when an intrusion occurs, and obtains information such as the download path and the secondary traffic after infection. Therefore, the results of the analysis are used to prevent intrusion in a more precise manner. Recently, due to the development of smartphone technology and the diversity of instant messaging (IM) software, defenders use packet sniffing techniques based on rule-based extraction methods to identify the IP addresses of cyber attackers [29,36–38].

An APT dedicated solution is similar to an existing security solution, but is APT-specific. This is an evolutionary solution that detects APT attacks that go undetected by existing security solutions using behavior-based analysis. It can be divided into host-based solutions and network-based solutions. A host-based solution has the advantage of detecting APT attacks by searching the host files and monitoring them in real time. However, it has a disadvantage in that it cannot detect unknown attacks. A network-based solution detects intrusions without affecting the real system by detecting and blocking malicious traffic by virtualizing inbound and outbound traffic to the network. This solution is advantageous in that it can easily conduct analyses by storing the detection and replay packets more accurately than existing security solutions by analyzing the secondary action. Nevertheless, since there is no distinction between clients in the organization, there are disadvantages in that it does not detect the spread of malware through correlation analysis, and it does not detect the spread through backdoor portals after penetration via USB. Researchers have recently studied many detection methods to predict the signs of APT attacks by analyzing the network traffic with network flow and combining it with clustering methods to overcome the shortcomings of these network-based solutions, and a combination of machine learning and deep learning models is used for higher detection rates [11,29,39,40].

The log analysis system responds to threats by analyzing logs of events in the system, detecting undesirable behaviors, and either informing the administrator about them or directly blocking them. These systems have the advantage of being more practical, because events that occur in the system result in various types of data, and various real-world situations can be derived from these. They include ESM (Enterprise Security Management), and SIEM (Security Information Event Management), which is an integrated security management system. It collects and manages logs generated by various security solutions and effectively manages and monitors these logs using analytical techniques such as correlation analysis, in-depth analysis, and association analysis. Thus, ESM makes it possible to detect and counteract various intrusions. SIEM is a combination of SIM and SEM; it manages and analyzes logs in a manner similar to ESM. However, unlike EMS, ESM takes a lot of time to process a large number of logs, as it must analyze short logs based on events and then process data based on a DBMS (Database Management System). SIEM was developed to overcome this issue of a prolonged processing time. It can analyze logs of data collected over a long period of time, as it uses big data processing technology, and it can do so quickly by using indexing-based processing [29].

The number of drive-by-download attacks, which are used to infect systems, has recently been increasing due to an increase in the rate at which web sites are accessed. Thus, there is a need for early detection and prevention of this type of attack. Various studies have attempted to develop such a method, including honeypots, which are widely used. A honeypot is a virtual system that analyzes and detects threats. It detects and blocks malicious codes by analyzing the behavior of the file that is transmitted from the web browser and communicating with the web server in the virtual system. Honeypots are classified as server-side honeypots and client-side honeypots depending on location. The server-side honeypot is a honeypot that waits for attacks by external attackers, while the client-side honeypot visits web sites through a crawler, collects web page codes and downloaded files, induces infection, and analyzes malicious behavior based on the collected information. Honeypots can also be classified into low-interaction client honeypots and high-interaction client honeypots. The former can operate in simple script form, so it does not require building a system, and it has the advantage of detecting known malicious behavior quickly. One such low-interaction client honey pot is Monkey-spider, which was developed by Mannheim University in Germany, which uses a Heritrix crawler to visit suspected URLs and detects malicious code by analyzing the source code of the downloaded website. However, low-interaction honeypots have a disadvantage in that they cannot analyze drive-by-download attacks. A high-interaction client honeypot can be used to build real-word systems to detect traffic on files that are downloaded after visiting a web site, monitor the movement of network traffic on path changes and the registry and file systems of a virtual environment, and monitor resource changes such as memory information. This type of honeypot has a disadvantage in that it takes time, because it performs analyses in greater detail than a low-interaction client honeypot. However, it can detect attacks that exploit vulnerabilities in application programs, because the system is implemented alone. For this reason, it is advantageous for the detection and prevention of unknown attacks. Therefore, defenders tend to prefer high-interaction client honeypots such as HoneyMonkey, Capture-HPC, etc. HoneyMonkey is a honeypot provided by Microsoft Corporation that visits suspicious URLs using Internet Explorer, monitors actions such as file and registry changes that occur during the execution of the generated file, and prevents the system from being infected by preventing the downloaded malicious program from running. Capture-HPC is a honeypot that uses VMWare, a virtual environment machine, to detect unusual changes by monitoring the state of the system by visiting suspicious URLs. In recent years, with the developments associated with the emergence of the Fourth Industrial Revolution, the use of Honeypot, which was previously limited to use with web pages, has also been used as a technology for cloud security. Moreover, research aiming to design smart agents for cybersecurity by incorporating machine learning technology is actively progressing [13,40–50].

The internal information leakage prevention technique prevents sensitive information stored in systems in the network from being leaked to the outside. This is performed by logging, searching for, and blocking important and confidential information by analyzing packets transmitted to the outside via services such as web mail, web hard, and P2P. However, the detection rate is low, and excessive detection occurs because it is difficult to define important information, so this technique is not used often for actual filtering purposes. Instead, it is used for post-tracking purposes through file archiving. Typical examples include SCM (Secure Content Management), OCC (Outbound Content Compliance), ILD (Information Leakage Detection and Prevention), and CMF (Content Monitoring and Filtering). SCM is an integrated content security solution that provides four security functions: anti-virus, anti-spyware, web filtering, and message security. OCC can provide the monitoring, encryption, filtering, and blocking of contents that are leaked to the outside through IM (Instant Messaging), P2P, and FTP. ILD&P minimizes the illegal leakage of sensitive information that is protected by legal measures such as HIPPA, GLBA, SOX, the Japanese personal information protection law, and other technical measures; its aim is to prevent the leakage of confidential information through IM and P2P. CMF detects and blocks the leakage of confidential information by analyzing traffic that is delivered to the inside and outside based on predefined rules and policies. These techniques control the leakage of sensitive information by modulating the access, transmission permission, contents, and formation of the data. However, it is difficult to detect the leakage of such information when the information is leaked using a benign service, or when it is compressed and encrypted so that the information is not visible [11,13,51].

Secure USB is a technique for preventing an insider attacker from leaking internal confidential information using a USB storage device. This protects internal information from being leaked to the outside by monitoring the registered USB storage devices in the internal network, using user authentication and encryption functions to prevent information leakage due to theft or loss, and using erase functions to prevent device disassembly. However, it has a limitation in that it cannot directly and intelligently detect certain dangers such as the presence of confidential information [11,52–54].

Systems are also protected through anti-spam filters and other similar functions. The main functions of a security control system are collecting information on security events such as network intrusion and system/network intrusion, and then reporting them in an integrated manner. Currently, security control systems utilize techniques that enable immediate response to the cyber threats that occur every moment through the risk classification of security events recorded by the security manager, normalization and rule-based event collection, and integrated policy management. However, since only the detected information is collected and analyzed by the interlocking security equipment, the detection capacity of a security control system depends on the strength of the equipment it uses. Moreover, due to the large number of security events that can accumulate in real time, there is not enough time to analyze the authenticity of each attack despite the fact that experts are in control. One false detection makes it more difficult to analyze the association between attacks over a long period of time. A security control system is an open and enterprise-wide integrated management platform based on multiple vendors that collects security events from each device by collaborating with security devices such as firewalls, web firewalls, and IDS, and it displays threat information according to predetermined patterns [26].

5.2. Pattern-Based Detection Method

Pattern-based detection methods have emerged because of the need to quickly inspect and analyze malicious codes coming from the outside. These methods detect and block a malicious code that matches patterns when the characteristics of known malicious codes are defined as specific patterns. They can be classified into signature-based detection, misuse detection, and malicious URL and IP detection according to the defined pattern.

The signature-based detection method detects malicious codes that match the characteristics of previously analyzed malicious codes with specific signatures. Depending on the type of signature, this method can be categorized into string signature, command frequency signature, control flow graph (CFG) signature, byte sequence signature, opcode occurrence frequency, and sequence signature techniques. The string signature uses ASCII code and Unicode as signatures in a file, and it defines a specific signature based on a string stored in the DLL file name, API name, data section, and text section. The command frequency signature uses the frequency of occurrence of commands found in existing malicious codes as a signature and defines the statistical distribution of the commands as a signature. However, its utility is limited by the amount of experimental data that are available. The control flow graph (CFG) signature represents the execution flow and structure of the code appearing in the code section of a file as a CFG graph and also uses it as a signature. A block is used to divide the sequence of the assembly code in the code section based on the branch instruction, and the technique recognizes a graph that appears as a vertex. The byte sequence signature generates sequences of bytes stored in a file and defines it as a signature. The N-gram technique is mainly used for this purpose. This technique defines the length of a byte sequence as N and then slides N sequences of each file for comparison. The opcode occurrence frequency and sequence signature are weighted based on the occurrence frequency of opcode in a suspicious file; an opcode with no significance and low weight by noise is removed, and malicious code is detected by using the cosine similarity based on the opcode sequence represented by the vector and frequency. As detailed above, these signature-based detection methods are advantageous in that they can quickly analyze and detect malicious codes by matching suspicious codes with similar portions of defined signatures. To elaborate, these methods are able to detect malicious codes very effectively, along with 50% of unknown malicious codes. Moreover, malicious codes which have been circulated in small quantities or with a special purpose are 20% detectable by Gartner, because more than 90% of the files that come into the network are conventional benign files and malicious code [12,13,22,28,55–62]. However, the signature method cannot detect unknown new malicious code due to the absence of a signature. This is because the signature method generates signatures based on the characteristics of the analysis results of the malicious code. Moreover, this method requires a lot of time to analyze malicious codes, along with more time to define and update the signature. Moreover, malicious codes that use zero-day vulnerabilities cannot be detected, and they are therefore difficult to respond to quickly [12,13,22,26,28,55–61,63–65].

Misuse detection is a method of detecting patterns corresponding to malicious code and malicious behavior that have already been detected, and this is accomplished by defining behavior rules and signatures as policies. This technique includes the signature analysis, expert system, state transition analysis, and Petri Net methods, and it has the advantages of a high detection rate and a fast detection speed, because malicious code is detected based on defined rules. However, as with the signature-based detection method, it is impossible to detect new and variant malicious codes without defined patterns and zero-day malicious codes that occur before rules are defined [22,26,63–65].

An attacker can spread malicious codes by penetrating vulnerable web servers and web sites, or by inducing a waypoint to spread malicious codes. Due to the active-X vulnerability in web browsers and vulnerabilities in JAVA and Adobe Flash, users can be infected with malicious codes through drive-by-download attacks when they access web servers and websites while remaining completely unaware of the infection. The infected system is used to conduct malicious actions such as theft of personal information, or used as a waypoint for the next attack. In particular, the attacker can hide code that accesses a web site such that it is not detected by a security program. These techniques manipulate the hidden attribute provided by a tag such as an iframe in the HTML code of the web page source file to prevent the user from recognizing the connection to the derived site. They also utilize encoding and code obfuscation techniques to block the intuitive awareness of string signatures in URLs and JavaScript code. In response, various measures have been taken to prevent access to malicious web servers and websites, and malicious URL and IP detection methods have been proposed. These methods compare the URL with the IP address of the currently connected site based on a directory of known malicious URL and IP addresses, and if they are found to match, the method identifies it as a harmful site, a phishing site, or an unnecessary site, and thus blocks the connection [12,13,29,66,67].

5.3. Heuristic-Based Detection Method

The pattern-based detection method has the advantage of enabling fast detection because it only compares defined signatures. However, there is a problem in that it cannot detect new malicious code or variant malicious code without signatures. The heuristic-based detection method for detecting variant malicious codes and new malicious codes was developed to solve this problem. This method identifies specific behaviors of malicious code, and it aids in the detection of variant malicious codes because it judges actions without comparing signatures. The method used to identify specific behaviors involves collecting commands related to file, registry, network communication, and resource usage, then determining the behavior of these commands as a heuristic signature and comparing existing and variant malicious codes with known malicious codes. Through this comparison of similarities, the heuristic detection method detects malicious code with simple changes, such as modifications of several bytes. However, as it is difficult to define the criteria for the comparison, benign programs are often identified as malicious code. The heuristic detection method can be classified into static heuristic, dynamic heuristic, negative heuristic, and generic algorithm methods [10,13,28,68–73].

The static heuristic detection method compares the heuristic signature of a file with a previously defined heuristic signature based on information in the file without having to execute the file [8,11,45,46]. The dynamic heuristic detection method defines and compares various information and behaviors that occur when a file is executed; this method is also referred to as run-time heuristic detection because files are detected as they are executed. The files are executed in a virtualized area or an area separate from the operating system to protect the system from infection during the detection process [10,13,70–72].

The negative heuristic detection method provides a way to define the heuristic signature of a benign file and allow for signature matching; this differs from the existing heuristic detection method of detecting and blocking malicious behavior. Since this is a whitelist-based detection method, it has the advantage of being able to detect new and variant malicious codes. However, it has a disadvantage in that malicious codes are detected as benign programs [28,74].

The generic algorithm-based method detects variants that are similar to malicious code that is already known. The degree of similarity is compared using a genetic algorithm based on instructions written in the code segment of a program; an opcode instruction comparison method is mainly used. This method can be classified as a wildcard or mismatch comparison method based on the means of comparison. The wildcard comparison method compares the degree of similarity by processing the wildcard portions of the opcode of a specific area while ignoring the portions that are changed. In this way, if malicious codes are slightly transformed or modified, it is possible to detect variant malicious codes because they have a high degree of similarity. Moreover, it is also advantageous in that there is no need for an update because an existing signature is used to compare the degree of similarity. However, when an obfuscation technique is used to prevent analysis of the instructions, a malicious code cannot be detected because the files do not appear similar, and the method cannot be used to detect a new (non-similar) malicious code. In recent years, Android malware detection methods applying genetic algorithms have mainly been studied, and Yildiz verified a malware detection rate of about 98% with a combination of genetic algorithms and support vector machines [10,28,75–78].

5.4. Reputation-Based Detection Method

To address the issues inherent to the signature-based detection method (i.e., the inability to detect variant and new malicious codes), user information such as suspicious files, behavior patterns, and characteristics was collected, and a reputation-based detection method that compares these data to an existing signature database was developed. The main goal of this method is to detect new malicious code. It uses feedback from a large number of users to determine the reliability of new files. The reputation information is defined in terms of the number of users and producers, etc., and the accuracy and reliability are determined according to whether or not reputation information is successfully obtained and analyzed. During the detection process, when a new file is found in the system, this method creates a unique value for the file using a hash function and transmits it to the server to inquire whether it is malicious. If it is malicious, a neutralization code is returned. When the eigenvalue of the new file is not retained, maliciousness is determined by behavior analysis, dynamic analysis, and malicious URL analysis of the transferred file, and the above process is executed based on the obtained result. It is therefore possible to detect malicious code quickly when a unique value is defined and stored. Therefore, when a new malicious code appears, it is identified instantly, and its reliability is judged according to its reputation, thus providing an improved detection method. However, this method has some problems. For example, an agent program must be installed in the system to collect reputation information; the method cannot be applied to detection technology that analyzes network traffic because the detection is carried out by considering files. Further, this method must be combined with other existing detection methods, such as signature-based and heuristic-based detection methods, to determine whether or not certain information is malicious because the method relies solely on reputation information. Further, due to the increasing number of applications and web browser plugins, this method cannot cope with various dynamic threats such as combinations of different types of vulnerability and the first emergence of an unknown malicious code. This reputation-based detection method can be classified into reputation-based detection and behavior-based detection methods [12,13,79].

The behavior-based detection method detects malicious behavior based on detailed information such as stack and context when a real action is performed instead of a single action. To overcome the problems of both false negatives and false positives, this method is often combined with the reputation-based method. This method detects malicious code that exploits vulnerabilities in document files such as HWP and PDF, as well as executable files, because detection is based on behavior. It also detects malicious codes that utilize web browser vulnerabilities, system patch files such as ws2help.dll, autorun commands that are propagated via USB, droppers, and other malicious codes that damage target systems. Further, when an unidentified process is found to be writing to the MBR area, writing to the root of the disk drive, or causing serious problems to the system such as by writing a host file, this method provides a function to confirm that the user is performing a requested behavior [12].

The reputation-based detection method defines the reputation information of a file based on the date of first discovery, the number of users, and the number of suspicious actions. When new files and processes are executed, this information is used to verify the reputation scores, and the user is warned about files and processes with low reputation scores. On the other hand, files and processes with high reputation scores are considered to be trusted and thus allowed to execute. In some cases, reputation scores are computed based on suspicious behavior. Some examples of suspicious behaviors include when an executed process deletes its own file or when processes are executed in an abnormal address range [12].

5.5. Behavior-Based Detection Method

The signature-based, heuristic-based, and reputation-based detection methods aim for malicious code detection. However, to apply these methods, information such as required signatures must be analyzed by malicious code analysts. There are also limitations to the ability of these methods to collect or analyze malicious code when the number of malicious codes increases, and when the speed at which malicious code is manufactured is much faster than the speed of analysis. It is also hard to analyze malicious codes realistically, because certain techniques are used to disturb the analysis of malicious code. Therefore, researchers have begun to analyze malicious behavior rather than analyzing the malicious file itself. The behavior-based detection method is a technique for detecting malicious behavior that occurs at the moment a file is executed. It represents an improvement upon the heuristic-based detection method. Malicious behavior occurs not only in executable files, but also in document files such as PDF, DOC, and HWP. The method determines the characteristics of malicious behavior according to files, registries, networks, and processes. Some examples of suspicious behavior include: when a DLL file and an .exe file are generated in a specific folder, such as a temporary folder that is not generated by a benign file; when a registry is registered for a specific service or a registry key value is altered to register startup programs; when a connection to a specific network is attempted; and when an executable file is executed to generate itself or an uncommon program such as a system program is executed. Thus, the detection of malicious activity can overcome the limitations of existing signature-based, heuristic-based, and reputation-based detection methods to detect new malicious code. Moreover, this method can use behavior to detect code variants and aid in deriving the necessary information that must be applied to existing detection methods. This is because an analyst can identify features such as the information and purpose of a malicious code, as well as its operation process in the system and propagation path. However, false negative and false positive results may be obtained as a result of irregular behavior from benign users, the use of avoidance techniques to disrupt the analysis, and the difficulty of detecting and responding at the kernel level.

This method can be classified into static, dynamic, and hybrid analysis methods according to the specific type of analysis that is applied. Static analysis is used to analyze behavior via reverse engineering using a disassembler and decompiler without the need for actual execution of the file, and to analyze the overall structure and flow of the code by examining the consistency of the code's component and call relationships. For such an analysis, when using a Windows operating system, a PE (Portable Executable) structure—which is an executable file format—is used to extract the elements of each section, and then the results are derived. To derive the information of the calling functions, the defender analyzes the flow of the operation and determines malicious behavior by tracking the control flow of blocks, which are API (Application Programming Interface) information analysis included in the IAT (Import Address Table); extracting strings; and dividing the code into smaller units. This method does not affect the analytical system, because it does not directly execute the file, and it is possible to analyze the structure and operation flow without being affected by the condition that the malicious code executes. However, there are problems with this method. For example, automation is troublesome because much of the analysis is manual, and it is therefore time- and labor-intensive; further, the analysis is difficult when a file is encrypted and packed. Dynamic analysis is a method of analyzing the characteristics that appear when a file is actually executed. This method involves detecting malicious behaviors by analyzing processes and files, registry creation and modification, network activities, and API call flow in an emulator and virtual machine in such a way as to not affect the actual system. When the emulator and the virtual machine are utilized, it is necessary to construct the environment and to prepare for API hooking in order to analyze the API call flow. Through API hooking, the situation can be monitored by collecting all of the information about API calling, and behaviors can be analyzed during execution by tracking the file, registry, process, network connection, etc. Therefore, this method is advantageous in that it is logically separated and does not affect the actual system, which can quickly be recovered to its pre-infection state even if the malicious code is executed, and the behavior can be accurately analyzed because the code is being executed at the time of analysis. However, analysis is difficult when malicious code is executed in a specific environment and under certain conditions, and only a part of the analysis can be performed, rather than analyzing all execution processes.

To solve the problems of static and dynamic analysis methods, a hybrid analysis method has emerged that solves the shortcomings of each analysis method and leverages the advantages of each one. The authors of [80] developed a framework for the detection of malicious code using both static and dynamic analysis methods. In addition, [81] uses both static and dynamic analysis methods, but only static analysis is performed in the test process whereas only dynamic analysis is used in the training process. By applying both the static and dynamic analysis methods as needed, the effectiveness of malicious code detection is increased [80–82].

Moreover, this method is classified into system-based and network-based detection methods according to the analytical targets of the behavior. The system-based detection method detects malicious behaviors based on the CPU, memory usage, and registry value and device ID that are generated when the program is executed. The network-based detection method analyzes network traffic and detects malicious behaviors stemming from network connections and transmitted packets. Specifically, the system-based detection method can be classified into the integrity check technique and behavior monitoring and blocking method based on the precise method of detecting malicious behaviors. There are also other methods that use API call behavior detection, visualization, behavior-dependent graphs, traffic characteristics, abnormal behavior detection, system call sequence state-based detection, and fuzzy logic and genetic algorithms [10,13,26,28,30,83–94].

As advancements continue to be made in malware analysis methods, malware creators use various obfuscation techniques, such as compression and encryption techniques, to hide specific signatures or malicious segments. Entropy-based malicious code analysis methods have been introduced to detect obfuscated malicious codes. Obfuscated malware, such as compressed and encrypted malware, has higher entropy than conventional malware. It is therefore possible to detect malicious codes due to the presence of high-entropy files. Recently, to improve the entropy-based malicious code analysis method, more reliable results were derived by introducing different entropy criteria and various entropy measurement methods for each file format, and the methods were verified to be models with optimal performance by applying various machine learning models [95,96]. However, entropy-based malicious code detection methods have several limitations. First, false positive detection of an actual encrypted file as a malicious code may occur. Second, the entropy of the plaintext for each file format is higher than that of the encrypted file. Third, it is difficult to calculate the entropy of files that have large sizes. Fourth, ciphertexts to which encoding methods such as Base64 are applied are limited in terms of detection, as the characteristics of ciphertexts disappear and entropy decreases. Fifth, false positives are to be found in compressed files in which features similar to those of ciphertexts appear. Finally, the application of partial encryption reduces entropy, which makes detection difficult, and there is a problem in that measuring the entropy of a pure file must precede each file format [97,98]. To address these problems, [99] proposed a method for the automatic quantification of entropy in files, [95,100] introduced a method for detecting entropy by dividing a file into multiple zones and a method for analyzing only the header of the file [95,96,99–103].

The integrity check technique checks the reliability of the data and executable files in a system. This method uses a hash function such as MD5 (Message Digest 5) to generate unique values for the executable files and folders of an uninfected system, then analyzes the behavior of malicious code by detecting changes in the values of existing files and folders or newly created files and folders by periodically comparing the eigenvalues. Therefore, when the eigenvalue of a file or folder is changed or not defined, the system and executed file are suspected of performing malicious actions. As a result, unknown new malicious codes can be detected. However, if the system is already infected, it is difficult to cope with and recover from the infection [10,28,83].

The behavior monitoring and blocking method monitors all activities that occur in a system, and blocks executable files by tracking the subject of the action when malicious actions occur. If a malicious action is detected, information about the file is displayed to the user, who can then decide whether or not to block the malicious action. In the case of a file executed by the user, the execution of the file is allowed to continue. Malicious code is detected and prevented by blocking the execution of files that are not executed by the user. In this method, malicious behaviors are determined based on API call processes and access objects, and studies have been conducted to compare the behavior graphs that are generated based on this information. The behavior monitoring and blocking method can be subdivided into the following categories: spear phishing email surveillance, frequency of occurrence of opcode sequence, common behavior graph showing malicious code execution behavior, framework based on machine learning methods, and association with registry and process information. Dewan proposed a spear phishing email surveillance method. This method combines the information contained in a social service profile with the information of an e-mail, including its title, contents, and attachments, to determine whether the e-mail is malicious, as e-mails represent one of the main paths through which malicious codes infect systems. This method detects spear phishing e-mails using various machine learning algorithms. Experimental results have shown that this method does not detect actual spear phishing e-mails, but it seems that such attempts are very helpful in detecting malicious codes. The frequency of occurrence of the opcode sequence method was proposed by Igor

Santos. This method detects malicious codes by comparing the frequency of occurrence of opcodes based on the relevance and frequency of opcodes in a code sequence, which is based on how often a sequence appears. The common behavior graph showing the malicious code execution behavior method was proposed by Park. This method creates a KOBG (Kernel Object Behavior Graph), which is a behavior graph that represents kernel objects and attributes them the nature of malicious codes that invoke system calls for attack. Then, malicious behaviors are detected by comparing the graph generated from malicious codes with a weighted behavior graph. The framework based on the machine learning method was proposed by Nir Nissimet. This method uses static analysis to represent malicious and benign executables. An association with registry and process information was proposed by Kim. This method analyzes the behavior patterns by examining the time, calling process, function, and registry key based on the characteristics of the registry and process information that is accessed by the malicious codes to perform malicious behaviors [10,28,71,76,86,89,90,104,105]. The API call behavior detection method generates threat indicators that are used to calculate a risk index; malicious behaviors are detected by analyzing the behavior of programs and files based on the threat indicators, which serve as indicators of correlation analysis. These threat indicators can be classified into risk, warning, and safety stages. Programs are installed and executed in the safety stage, but they are blocked in the dangerous stage [11].

Visualizations can be classified into tree map-based, grayscale of binary content, function call graph, control flow graph, and behavior-dependent graph visualizations. Trinius proposed tree map-based visualization, which identifies malicious behavior and classifies malicious targets into processes, files, and registers. When the same behavior is detected by comparing tree maps, malicious code is detected [13,106]. The grayscale-based visualization of binary content was proposed by Nataraj. Binary contents that appear in malicious code are visualized as grayscale images, and malicious code is determined based on the degree of visual similarity [107–109]. A function call graph (FCG) is a graphical representation of functions contained within a program, which is a directional graph on which the edge indicates a function call. In addition, Hanssen and Chan recently proposed a malware detection method that extracts vectors from function-call graphs based on function clustering. A control flow graph (CFG) was proposed by Eskandari. This method detects variant malware using a control flow graph [13,110].

The behavior-dependent graph was proposed by Fredrikson. This method detects malicious codes by comparing existing behavior-dependent graphs with the behavior graph extracted from an incoming file. In another study, the traffic characteristic method was proposed by Xuzian. This method defines the behaviors of traffic generated by malicious codes by detecting different parts of the traffic characteristics between benign traffic and malicious traffic [13,111]. The abnormal behavior detection method generates a model of benign behavior, then determines malicious behaviors when observing actions that have not been included in the model. Data mining and statistical methods are used to generate models of benign behaviors [22,112,113]. The system call sequence state-based detection method analyzes benign behavior and abnormal behavior by examining the patterns of the correlation of states based on a sequence of system calls. This method defines a sequence of system calls that are called during a specific time period as a state, and it detects malicious codes based on patterns that indicate transitions of the classified state transitions. These can be divided into eight categories (file system, kernel, memory management, and so on) [22,114]. The use of fuzzy logic and genetic algorithms was proposed by Kaur. This method detects abnormal behaviors by defining the actions of a benign user by applying fuzzy logic to system log files. It is possible to generate new user behavior rules by applying a genetic algorithm to optimize user behaviors [22,115].

5.6. Virtualization-Based Detection Method

The behavior-based detection method detects malicious behaviors involving target files, such as executable files and document files. However, the system can be infected in this process because it involves executing the file on the actual system. To solve this problem, it is necessary to create an environment that is separate from the system in which malicious behaviors can be analyzed in a safe space. A virtualization-based detection method has emerged to accomplish this. This method is closely related to the dynamic heuristic detection method. The virtualization-based detection method tracks malicious behaviors by monitoring the behavioral changes that occur when a file is executed in the same manner as the behavior-based detection method. The main difference is that it runs in an environment that is physically and logically isolated from the actual system. In this way, it is possible to protect the system from actions caused by untrusted programs, and the system is neither infected by malicious codes nor propagates such codes to the outside. Therefore, the characteristics of this method are very similar to those of the behavior-based detection method. However, the two differ because the former method makes it possible to construct a virtual space in an actual system to obtain execution and control information and to examine executable files. Moreover, a study was conducted in parallel with static analysis to achieve more reliable detection. Specifically, the tint analysis method, which analyzes the flow and purpose of data used in the process of execution by inserting the code to be analyzed in the middle of the execution code, was used instead of simply analyzing the code flow and behavior. This method is classified into an emulator and a sandbox method depending on how the virtual environment is constructed [10,13,28,85,86,116–121].

The emulator, which is an alternate launcher, is used to indirectly execute a file via virtual hardware based on a virtual environment and a virtual hardware image; this is also referred to as code emulation or CPU emulation. This method is advantageous in that the actual system is not affected, and the method can effectively detect unknown malicious codes due to the above characteristics [10,86]. This method analyzes the behavior that occurs when the file is executed based on the emulation controller that controls the virtual environment supporting the CPU and memory in the system.

The emulator creates a virtual space, but the sandbox collects information by running the file on the real system and detects malicious behaviors by monitoring information related to the activities of a process, a file, and a network. This method leaves the system vulnerable to infection by malicious codes because it involves executing the file in the actual system. Nevertheless, the system is not practically compromised by infection because the calling process is controlled by a specific tool. Representative products and systems include CWSandbox, Norman Sandbox, and Anubis. CWSandbox injects VMMonitor.dll, a monitoring module, into the thread of the file to be executed, then traces all of the procedures that are executed in the module and passes this information to CWSandbox. Accordingly, CWSandbox prevents the system from being infected by transmitting a benign return value to the operating system if the code is benign. Any malicious code that is detected is instead filtered. Despite these advantages, attackers have created new malicious codes to prevent detection in a virtualized environment, while sandbox services based on free and open-source software, such as CWSandbox, Zerowine, and Cuckoo Sandbox, have become more generalized. Hence, the defender cannot cope with this problem. Moreover, malicious codes are not detected when techniques for bypassing detection of malicious code are used; examples of such techniques include the human interaction technique, which only operates when performing specific actions; the anti-debugging technique, which does not perform malicious actions when the tool that analyzes malicious code is executed; and the anti-virtual machine technique, which does not perform malicious actions when executed in a virtual environment. Detection can also be bypassed if the malicious code is packed, if the malicious code has a built-in latency period that makes it wait until it has passed through the sandbox, and if the malicious code has a time trigger that causes it to run only on certain dates and times [10,13,85,118–121].

The machine learning-based malicious code analysis method is similar to the big data-based detection method. It detects malicious codes based on features that appear in a large amount of data. The biggest difference between the two methods is that the former learns a model. Machine learning-based malware detection methods have emerged to address the fundamental problems associated with traditional malware detection methods, such as security systems, pattern-based detection, heuristic-based detection, reputation-based detection, behavior-based detection, virtualization-based detection, anomaly detection, and big data-based detection. The known problems with existing methods include the inability to detect unknown new and variant malware and difficulty in detecting and counteracting zero-day attacks in pattern-based detection; high false positive and false negative rates caused by difficulty defining similarity criteria in heuristic-based detection; the need to install an additional program to obtain reputation, the difficulty of detection using network traffic analysis, and the requirement for fundamental combinations with other detection methods due to high false positive and false negative rates in reputation-based detection; the need to spend a lot of time and effort on manual analysis of malicious code and difficulty analyzing malicious code that is encrypted or packed in static analysis; difficulty with malicious code running in specific environments and conditions in dynamic analysis; the need for additional resources to reduce the false positive and false negative rates and unsuitability for use with unstructured data in anomaly detection; and dependence on the validity and size of the data in big data-based detection. In attempts to overcome these limitations, various malicious code detection methods based on machine learning have been researched. Machine learning can be classified into supervised learning, unsupervised learning, and reinforcement learning according to the learning method, and various models are used for learning [73]. Supervised learning is used to infer a function from training data that is used in regression analysis to predict the functional relationship between different data classes. Unsupervised learning differs from supervised learning, where the target value of the input value is not given. Reinforcement learning is a method of choosing the action or sequence of actions that provides the most reward among various options.

Various machine learning models have emerged, as described above, and they can be used in conjunction with existing malicious code detection methods, such as security control system, pattern-based malicious code detection, and heuristic-based detection. The recent trend in malware detection and prevention research is not to propose new detection methods, but rather to apply machine learning algorithms to existing malware detection methods. Hence, future research examining malicious code detection is expected to produce more effective methods of detecting and preventing malware using machine learning models [9,122–127].

5.7. Anomalous Symptom Detection Method

Existing pattern-based, heuristic-based, reputation-based, behavior-based, and virtualization-based detection methods can detect and prevent malicious codes. However, existing methods cannot adequately defend against penetrating attacks that utilize unknown attack techniques, such as APT attacks, as these methods require a lot of time to collect various types of information. To solve these problems, a method for detecting anomalous symptoms in the system has been proposed. This method uses correlation analysis to detect anomalous symptoms by collecting and integrating logs of activity in the system. In particular, this requires detection techniques at a stage where the system is infected by malicious code, the detection of unknown new malicious code, and techniques to determine whether traffic is benign or abnormal. To achieve this, techniques for analyzing network behavior and state have emerged, as have intelligent security technologies. The technique for analyzing network behavior is a method of detecting malicious behavior based on the behavior of a large-capacity and real-time network of various systems, protocols, and users. The technique for analyzing network state is a method of analyzing malicious behavior based on the system of the presently connected or requesting network and its state changes. Modern intelligent security technology analyzes threats in real time by integrating these types of information and methods. Therefore, certain information must be collected before this method can be used. The characteristics of this information are defined by applying statistical techniques to data, such as a profiling technique for the extraction of meaningful information and a data mining technique for the extraction of statistical rule patterns from a large amount of data. Nevertheless, despite these advantages, it is technically difficult to collect large-capacity and high-speed traffic from a network or to analyze the correlations in this traffic, because the system is exposed to threats if the security system is not capable of real-time processing [13].

5.8. Data Analysis-Based Detection

Data analysis-based detection detects malicious codes based on features that appear in a large amount of data. This type of detection can be sub-classified into a big data-based detection method and a machine learning-based detection method according to the data analysis method.

To detect anomalous behaviors, big data technology is used to detect malicious activities based on massive amounts of network traffic and logs. As a result, big data technology, which is capable of collecting and analyzing a large number of logs, has been combined with security technologies in earnest. Big data technology prevents and predicts threats based on logs and traffic. Some programs that have been used for this purpose include Hadoop, Map Reduce, and R as an open-source option. This method differs from the anomalous symptom detection method, as it allows for not only effective and quick analysis through the reprocessing of logs, but also intuitive determination of the results through visualization. Detection methods using big data can be classified into real-time monitoring, intelligence on threats, behavior profiling, data and user monitoring, application monitoring, and analysis and interpretation methods. The real-time monitoring technique involves tracking attacks on system components and monitoring user activities in application programs. The intelligence on threats technique intelligently detects various threats and attack patterns to accurately recognize even small amounts of malicious traffic that has been disguised as benign traffic. Behavior profiling is used to detect anomalous symptoms by defining association rules based on well-defined abnormal conditions. The data and user monitoring technique detects when sensitive information and privileged user profiles are being accessed by monitoring user activity and data. The application monitoring technique monitors the activity of abnormal applications. The analysis and interpretation technique determines abnormal behaviors from the characteristics of information collected from various sources using machine learning, data mining, and network mining techniques [23,29,128–130].

5.9. Other Detection Methods

Other detection methods (i.e., additional measures to detect malicious codes) include the packer detection and unpacking techniques as well as the malware-clustering technique. The packer detection and unpacking techniques detect and unpack a packer to neutralize a packing technique that modifies or encrypts code to prevent malicious code from being analyzed. The malware-clustering technique analyzes the correlation between a large number of malicious codes to detect variant malicious codes, and it collects information using information calls for malicious actions. Recently, Mi-Jung Choi [131] proposed an all-in-one framework that includes packer detection, unpacking technology, and malicious code detection all together [13,131].

5.10. Analysis of Advantages and Disadvantages of Malicious Code Detection Methods

Based on the results of the above analysis, we compare and analyze the advantages and disadvantages of different malicious code detection methods; the results are presented in Table 4. The O, Δ, and X symbols indicate whether malicious code types—such as analyzed malicious code, new malicious code, and variable malicious code and packing or encryption—can be detected by each malicious code detection method (the O symbol means it is detectable, X is not detectable, and Δ is detectable, but depending on the data, the detection accuracy is relatively low or difficult).

Table 4. Advantages and disadvantages of various malicious code detection methods. (O: detectable, X: Undetectable, Δ: Dependent).

Methods	**Information for Detection**	**Malicious Code Types**				**Detection Characteristics**			**Problems**
		Analyzed Malicious Code	**New Malicious Code**	**Variant Malicious Code**	**Packing or Encryption**	**Detection Speed**	**Detection Rate**	**False Negative and False Positive**	
Pattern	Signature (string, command, sequence, graph)	O	X	X	X	Fast	Analyzed malicious code: High New malicious code: Low Variant malicious code: Low	Low	Update rules and policies required pre-analysis
Heuristic	Behavior (file, registry, network communication, resource usage)	O	X	O	X	Fast	Analyzed malicious code: High New malicious code: Low Variant malicious code: High	High	Need for definition of similarity comparison Require pre-analysis
Reputation	Reputation (date of first discovery, number of users, number of suspicious actions)	O	Δ	Δ	X	Fast	Analyzed malicious code: High New malicious code: Low Variant malicious code: Medium	High	Need for agent installation of reputation collection Not applicable to network traffic
Behavior	Behavior (system, network, call flow and correlation)	O	O	O	Δ	Late	Analyzed malicious code: High New malicious code: High Variant malicious code: High	High	Difficult to detect applied anti-debugging and anti-virtual machine techniques Difficult to detect APT attack; high cost and time spent transforming data
	Visual information (tree map, binary contents, FCG, CFG, BDG)	O	O	O	Δ	Late	Analyzed malicious code: High New malicious code: High Variant malicious code: High	Medium	
Anomalous symptom	Log (correlation)	O	O	O	X	Late	Analyzed malicious code: High New malicious code: High Variant malicious code: High	Medium	Exposed to threats when real-time processing is disabled Difficult to analyze and detect in high-speed traffic
Big data	Traffic and Log (probability, statistics)	Δ	Δ	Δ	Δ	Late	Analyzed malicious code: Medium New malicious code: Medium Variant malicious code: Medium	Medium	Requires a lot of time for learning data Performance difference according to the quantity and quality of training data
Machine learning	All information (signature, behavior, visual data, reputation, log, traffic)	O	O	O	O	Fast	Analyzed malicious code: High New malicious code: High Variant malicious code: High	Low	

We find that most detection methods cannot detect packed or encrypted malicious code. The behavior-based detection method requires unpacking and decryption. All detection methods detect known malicious codes. However, the pattern-based detection method cannot detect new malicious codes and variant malicious codes, because it is based on patterns that have already been found in known malicious codes. The heuristic-based detection method also has a limitation in that it cannot detect new malicious codes, as is the case with the pattern-based detection method. The reputation-based detection method and the big data-based detection method (two data analysis-based methods) cannot detect new malicious codes and variant malicious codes either. This is because the reputation-based detection method identifies malicious code based on the reputation collected from users. For this reason, if incorrect reputation information is included, then this method will not be able to detect new malicious codes. The big data-based detection method and machine learning-based detection method (two data analysis-based methods) rely on the data used for learning. For this reason, detection is difficult when there is insufficient data or when features cannot be extracted to detect new malicious codes. In terms of detection rate, the pattern-based detection method, heuristic-based detection method, and reputation-based detection method are all fast, because they simply compare specific patterns. Meanwhile, the conventional behavior-based, visualization-based, anomalous symptom, big data-based, and machine learning-based detection methods are slower. This is because the behavior-based detection method and anomalous symptom detection method require time to determine malicious behavior, while the big data-based detection method requires time for learning. The various detection methods differ in terms of detection rate and false negative and false positive rates because of the different characteristics of each method. In particular, among the detection methods based on data analysis, the machine learning-based detection method has more severe deviations in detection rate due to false positive and false negative results depending on the quantity and quality of training data, and it requires a large amount of information and time for initial learning. However, when the initial learning is completed, this method has a very fast detection speed and detects known, new, and variant malicious codes, and it can be applied to detect all kinds of malicious codes, such as through pattern, behavior, visualization, reputation, and anomalous symptom analysis.

6. Conclusions

In this paper, we analyzed attack scenarios based on attack cases to detect penetrating malicious codes via network and system, and we surveyed and analyzed attack techniques used in real attack scenarios. Based on the results, we classified malicious code detection techniques into security control system, pattern-based, heuristic-based, reputation-based, behavior-based, virtualization-based, and anomalous symptom detection methods, and we classified big data-based and machine learning-based detection methods into data analysis-based methods. Each malicious code detection method has its own pros and cons, as well as unique limitations. The defender should employ a proper combination that considers the system and the network to which each method is applied based on the characteristics and environment of the system. The malicious code detection method classification and analysis in this paper is expected to provide a basis for further research into the detection and prevention of malicious codes.

Author Contributions: Conceptualization, K.Y.; methodology, K.L. and K.Y.; data curation, K.L.; writing—original draft preparation, K.L.; writing—review and editing, J.L. and K.Y.; project administration, K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was mainly (70%) funded by a grant (No. NRF-2021R1F1A1050542) of the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT). This work was also partly (20%) supported by a grant (No. NRF-2021R1A4A2001810) of the NRF funded by the Korea government (MSIT) and the Soonchunhyang University Research Fund (10%).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are contained within the article.

Acknowledgments: This research was mainly (70%) funded by a grant (No. NRF-2021R1F1A1050542) of the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT). This work was also partly (20%) supported by a grant (No. NRF-2021R1A4A2001810) of the NRF funded by the Korea government (MSIT) and the Soonchunhyang University Research Fund (10%).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khalid, A.; Zainal, A.; Maarof, M.A.; Ghaleb, F.A. Advanced Persistent Threat Detection: A Survey. In Proceedings of the 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 5 April 2021; pp. 1–6.
2. Saurabh, S.; Sharma, P.; Yeon Moon, S.; Daesung, M.; Hyuk Park, J. A comprehensive study on APT attacks and countermeasures for future networks and communications: Challenges and solutions. *J. Supercomput.* **2016**, *75*, 4543–4574.
3. Joint Task Force Transformation Initiative. In *Managing Information Security Risk: Organization, Mission, and Information System View*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2011; p. NIST SP 800-39.
4. Kim, G.; Choi, C.; Choi, J. Ontology Modeling for APT Attack Detection in an IoT-Based Power System. In Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, Honolulu, Hawaii, 9 October 2018; pp. 160–164.
5. Fei, Y.; Ning, J.; Jiang, W. A Quantifiable Attack-Defense Trees Model for APT Attack. In Proceedings of the 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 October 2018; pp. 2303–2306.
6. Gao, P.; Xiao, X.; Li, Z.; Xu, F.; Kulkarni, S.R.; Mittal, P. AIQL: Enabling Efficient Attack Investigation from System Monitoring Data. In Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference, Boston, MA, USA, July 2018; pp. 113–125.
7. Chuan, B.L.J.; Singh, M.M.; Shariff, A.R.M. APTGuard: Advanced Persistent Threat (APT) Detections and Predictions Using Android Smartphone. In *Computational Science and Technology*; Alfred, R., Lim, Y., Ibrahim, A.A.A., Anthony, P., Eds.; Springer: Singapore, 2019; Volume 481, pp. 545–555.
8. Nicho, M.; Oluwasegun, A.; Kamoun, F. Identifying Vulnerabilities in APT Attacks: A Simulated Approach. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–4.
9. FireEye, M-Trends. 2022. Available online: https://content.fireeye.com/m-trends/rpt-m-trends-2022. (accessed on 5 January 2023).
10. Esfahani, A.; Mantas, G.; Barcelos, M.; Saghezchi, F.B.; Sucasas, V.; Bastos, J.; Rodriguez, J. Security Framework for the Semiconductor Supply Chain Environment. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; Victory, S., Georggios, M., Eds.; Springer: Cham, Switzerland, 2019; Volume 263, pp. 159–168.
11. Kim, N.; Lee, S.; Cho, H.; Kim, B.-I.; Jun, M. Design of a Cyber Threat Information Collection System for Cyber Attack Correlation. In Proceedings of the 2018 International Conference on Platform Technology and Service (PlatCon), Jeju, Republic of Korea, 29–31 January 2018; pp. 1–6.
12. Moon, D.; Pan, S.B.; Kim, I. Host-Based Intrusion Detection System for Secure Human-Centric Computing. *J Supercomput* **2016**, *72*, 2520–2536. [CrossRef]
13. Kim, Y.; Kim, I. Involvers Behavior-based Modeling in Cyber Targeted Attack. In Proceedings of the Eighth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE), Lisbon, Portugal, 16–20 November 2014.
14. Jonathan, A.P.; Kim, K.H.; Park, J.-H.; Kim, C.-H.; Lee, H.-J. Analysis of the 2013.3.20 South Korea APT Attack. In Proceedings of the Korean Institute of Information and Commutation Sciences Conference, Mokpo, Republic of Korea, 24–25 May 2013; pp. 249–252.
15. Park, D.-W. Preemptive Response Strategy for Attacker Origin for National Cybersecurity. *Int. Inf. Inst. Inf.* **2018**, *21*, 277–285.
16. Kim, Y.S.; Kim, I.; Park, N. Analysis of cyber attacks and security intelligence. In *Mobile, Ubiquitous, and Intelligent Computing*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 489–494.
17. Agham, V. Unified threat management. *Int. Res. J. Eng. Technol.* **2016**, *3*, 32–36.
18. McAfee, Gold Dragon Widens Olympics Malware Attacks, Gains Permanent Presence on Victims' Systems. 2018. Available online: https://vxug.fakedoma.in/archive/APTs/2018/2018.02.02/Gold%20Dragon.pdf. (accessed on 25 January 2023).
19. Sharma, R.; Arya, R. Secure transmission technique for data in IoT edge computing infrastructure. *Complex Intell. Syst. Nov.* **2021**, *8*, pp. 3817–3832. [CrossRef]
20. Swain, S.C. Cybersecurity Threats and Technology Adoption in the Indian Banking Sector: A Study of Retail Banking Customers of Bhubaneswar. In *Strategies for e-Service, e-Governance, and Cybersecurity: Challenges and Solutions for Efficiency and Sustainability*; CRC Press: Boca Raton, FL, USA, 2021.
21. Vardalaki, A.; Vlachos, V. Emerging Malware Threats: The Case of Ransomware. In *Cybersecurity Issues in Emerging Technologies*; CRC Press: Boca Raton, FL, USA, 2021; pp. 153–170.

22. Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Aparicio-Navarro, F.J.; Rabie, K.; Jabban, A. Disguised executable files in spear-phishing emails: Detecting the point of entry in advanced persistent threat. In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (ICFNDS), New York, NY, USA, 26–27 June 2018.
23. Heartfield, R.; Loukas, G. Protection against semantic social engineering attacks. In *Versatile Cybersecurity*; Maruro, C., Gaurav, S., Eds.; Springer: Cham, Switzerland, 2018; Volume 72, pp. 99–140.
24. Kim, J. North Korea's cyber attack threat analysis research(Based on the type of attack technology). In Proceedings of the Korean Society of Computer Information Conference, Jeju, South Korea, 16–18 July 2020; pp. 107–110.
25. Rath, S.; Zografopoulos, I.; Konstantinou, C. Stealthy Rootkit Attacks on Cyber-Physical Microgrids: Poster. In Proceedings of the Twelfth ACM International Conference on Future Energy Systems, Virtual Event, Italy, 22 June 2021; pp. 294–295.
26. Li, M.; Huang, W.; Wang, Y.; Fan, W.; Li, J. The Study of APT Attack Stage Model. In Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, Japan, 26–29 June 2016; pp. 1–5.
27. Bermejo Higuera, J.; Abad Aramburu, C.; Bermejo Higuera, J.-R.; Sicilia Urban, M.A.; Sicilia Montalvo, J.A. Systematic Approach to Malware Analysis (SAMA). *Appl. Sci.* **2020**, *10*, 1360. [CrossRef]
28. Tan, C.; Wang, Q.; Wang, L.; Zhao, L. Attack Provenance Tracing in Cyberspace: Solutions, Challenges and Future Directions. *IEEE Netw.* **2019**, *33*, 174–180. [CrossRef]
29. Milajerdi, S.; Eshete, B.; Gjomemo, R.; Venkatakrishnan, V.N. ProPatrol: Attack Investigation via Extracted High-Level Tasks. In *Information Systems Security*; Ganapathy, V., Jaeger, T., Shyamasundar, R.K., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 11281, pp. 107–126, ISBN 978-3-030-05170-9.
30. Saleem, J.; Hammoudeh, M. Defense Methods Against Social Engineering Attacks. In *Computer and Network Security Essentials*; Daimi, K., Ed.; Springer International Publishing: Cham, Switzerland, 2018; pp. 603–618, ISBN 978-3-319-58423-2.
31. Faruki, P.; Laxmi, V.; Gaur, M.S.; Vinod, P. Behavioural Detection with API Call-Grams to Identify Malicious PE Files. In Proceedings of the First International Conference on Security of Internet of Things-SecurIT '12, Kollam, India; 2012; pp. 85–91.
32. Shahzad, F.; Shahzad, M.; Farooq, M. In-Execution Dynamic Malware Analysis and Detection by Mining Information in Process Control Blocks of Linux OS. *Inf. Sci.* **2013**, *231*, 45–63. [CrossRef]
33. Sengan, S.; Khalaf, O.I.; Sharma, D.K.; Hamad, A.A. Secured and privacy-based IDS for healthcare systems on E-medical data using machine learning approach. *Int. J. Reliab. Qual. Healthcare* **2022**, *11*, 1–11. [CrossRef]
34. Mehmood, M.; Javed, T.; Nebhen, J.; Abbas, S.; Abid, R.; Bojja, G.R.; Rizwan, M. A hybrid approach for network intrusion detection. *Comput. Mater. Contin.* **2021**, *70*, 91–107. [CrossRef]
35. Otoum, Y.; Liu, D.; Nayak, A. DL-IDS: A deep learning–based intrusion detection framework for securing IoT. *Trans. Emerg. Telecommun. Technol.* **2019**, *33*, e3803. [CrossRef]
36. Ahmed, W.; Shahzad, F.; Javed, A.R.; Iqbal, F.; Ali, L. WhatsApp Network Forensics: Discovering the IP Addresses of Suspects. In Proceedings of the 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 19 April 2021; pp. 1–7.
37. Qureshi, S.; Tunio, S.; Akhtar, F.; Wajahat, A.; Nazir, A.; Ullah, F. Network Forensics: A Comprehensive Review of Tools and Techniques. *IJACSA* **2021**, *12*. [CrossRef]
38. Son, K.H.; Tajine, L.; Won, D. Design for Zombie PCs and APT Attack Detection based on traffic analysis. *J. Korea Inst. Inf. Secur. Cryptol.* **2014**, *24*, 491–498. [CrossRef]
39. Zhao, G.; Xu, K.; Xu, L.; Wu, B. Detecting APT Malware Infections Based on Malicious DNS and Traffic Analysis. *IEEE Access* **2015**, *3*, 1132–1142. [CrossRef]
40. Negi, P.S.; Garg, A.; Lal, R. Intrusion detection and prevention using honeypot network for cloud security. In Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 January 2020.
41. Sun, X.; Wang, Y.; Ren, J.; Zhu, Y.; Liu, S. Collecting Internet Malware Based on Client-Side Honeypot. In Proceedings of the 2008 the 9th International Conference for Young Computer Scientists, 18–21 November 2008; pp. 1493–1498.
42. Zhuge, J.; Holz, T.; Han, X.; Song, C.; Zou, W. Collecting Autonom s Spreading Malware Using High-Interaction Honeypots. In *Information and Communications Security*; Qing, S., Imai, H., Wang, G., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4861, pp. 438–451.
43. Wang, Y.-M.; Beck, D.; Jiang, X.; Roussev, R.; Verbowski, C.; Chen, S.; King, S. Automated Web Patrol with Strider HoneyMonkeys. In Proceedings of the 2006 Network and Distributed System Security Symposium, San Diego, CA, USA, 3–5 December 2006; pp. 35–39.
44. Radek, H. *The Capture-HPC Client Architecture*; Technical report; Victoria University of Wellington: Wellington, New Zealand, 2009.
45. Ikinci, A.; Holz, T.; Freiling, F. Monkey-spider: Detecting malicious websites with low-interaction honeyclients. *Sicherheit* **2008**, *8*, 407–421.
46. Mohr, G.; Stack, M.; Ranitovic, I.; Avery, D.; Kimpton, M. An Introduction to Heritrix. In Proceedings of the 4th International Web Archiving Workshop, Bath, UK, 16 September 2004; Volume 15, pp. 109–115.
47. Xie, M.; Wu, Z.; Wang, H. HoneyIM: Fast Detection and Suppression of Instant Messaging Malware in Enterprise-Like Networks. In Proceedings of the Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007; pp. 64–73.
48. Xie, M.; Wu, Z.; Wang, H. Secure Instant Messaging in Enterprise-like Networks. *Comput. Netw.* **2012**, *56*, 448–461. [CrossRef]

49. El Kamel, N.; Eddabbah, M.; Lmoumen, Y.; Touahni, R. A Smart Agent Design for Cyber Security Based on Honeypot and Machine Learning. *Secur. Commun. Netw.* **2020**, *2020*, 1–9. [CrossRef]
50. Devi, B.T.; Shitharth, S.; Jabbar, M.A. An Appraisal over Intrusion Detection Systems in Cloud Computing Security Attacks. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 722–727.
51. Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Han, L.; Raza, U. Malicious SSL Certificate Detection: A Step Towards Advanced Persistent Threat Defence. In Proceedings of the Proceedings of the International Conference on Future Networks and Distributed Systems, Cambridge, UK, 19 July 2017.
52. Lee, K.; Oh, I.; Lee, Y.; Lee, H.; Yim, K.; Seo, J. A Study on a Secure USB Mechanism That Prevents the Exposure of Authentication Information for Smart Human Care Services. *J. Sens.* **2018**, *2018*, 2089626. [CrossRef]
53. Jung, W.; Yim, K.; Lee, K. Vulnerability Analysis of a Secure USB Memory: Based on a Commercial Product D. In *Advances on Broad-Band Wireless Computing, Communication and Applications*; Barolli, L., Ed.; Lecture Notes in Networks and Systems; Springer International Publishing: Cham, Switzerland, 2022; Volume 346, pp. 279–283.
54. Lee, J.D.M. FingerTool v1.19 in the DM PD065 Secure USB is Susceptible to Improper Authentication by a Replay Attack. The MITRE Corporation. Available online: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-26824 (accessed on 23 December 2021).
55. Firstbrook, P.; MacDonald, N. A buyers guide to endpoint protection platforms. Available online: https://www.gartner.com/doc/2973617/buyers-guide-endpoint-protection-platform (accessed on 23 December 2021).
56. Jung, B.; Kim, T.; Im, E.G. Malware Classification Using Byte Sequence Information. In Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems, Honolulu, Hawaii, 9 October 2018; pp. 143–148.
57. Tian, R.; Batten, L.; Islam, R.; Versteeg, S. An Automated Classification System Based on the Strings of Trojan and Virus Families. In Proceedings of the 2009 4th International Conference on Malicious and Unwanted Software (MALWARE), Montreal, QC, USA, 13–14 October 2009; pp. 23–30.
58. Bilar, D. Opcodes as Predictor for Malware. *IJESDF* **2007**, *1*, 156. [CrossRef]
59. Bonfante, G.; Kaczmarek, M.; Marion, J.-Y. Morphological Detection of Malware. In Proceedings of the 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE), Fairfax, VI, USA, 13 October 2008; pp. 1–8.
60. Pektaş, A.; Eriş, M.; Acarman, T. Proposal of N-Gram Based Algorithm for Malware Classification. In Proceedings of the the Fifth International Conference on Emerging Security Information, Systems and Technologies, Nice/Saint Laurent du Var, France, 21–27 August 2011; pp. 7–13.
61. Santos, I.; Brezo, F.; Nieves, J.; Penya, Y.K.; Sanz, B.; Laorden, C.; Bringas, P.G. Idea: Opcode-Sequence-Based Malware Detection. In *Engineering Secure Software and Systems*; Massacci, F., Wallach, D., Zannone, N., Eds.; Lecture Notes in Computer Science; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2010; Volume 5965, pp. 35–43. ISBN 978-3-642-11746-6.
62. Liu, Z.; Japkowicz, N.; Wang, R.; Cai, Y.; Tang, D.; Cai, X. A Statistical Pattern Based Feature Extraction Method on System Call Traces for Anomaly Detection. *Inf. Softw. Technol.* **2020**, *126*, 106348. [CrossRef]
63. Zhou, L.; Liu, F. Research on Computer Network Security Based on Pattern Recognition. In Proceedings of the SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No.03CH37483), Washington, DC, USA, 8 October 2003; Volume 2, pp. 1278–1283.
64. Li, P.; Teng, W.-D.; Zheng, W.; Zhang, K.-H. Formalized Answer Extraction Technology Based on Pattern Learning. In Proceedings of the International Forum on Strategic Technology 2010, Ulsan, Republic of Korea, 13–15 October 2010; pp. 236–240.
65. Shreeranga, P.R.; Vig, A.; Narayana, V.S.A. An Efficient Classification Algorithm Based on Pattern Range Tree Prototypes. In Proceedings of the 10th International Conference on Information Technology (ICIT 2007), Rourkela, Orissa, India, 17–20 December 2007; pp. 50–55.
66. Egele, M.; Kirda, E.; Kruegel, C. Mitigating Drive-By Download Attacks: Challenges and Open Problems. In *iNetSec 2009–Open Research Problems in Network Security*; Camenisch, J., Kesdogan, D., Eds.; IFIP Advances in Information and Communication Technology; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2009; Volume 309, pp. 52–62, ISBN 978-3-642-05436-5.
67. Niki, A. Drive-by Download Attacks: Effects and Detection Methods. Ph.D. thesis, Master's thesis, Royal Holloway University of London, London, UK, 2009.
68. Zhou, R.; Pan, J.; Tan, X.; Xi, H. Application of CLIPS Expert System to Malware Detection System. In Proceedings of the 2008 International Conference on Computational Intelligence and Security, Suzhou, China, 13–17 December 2008; pp. 309–314.
69. Al Daoud, E.; Jebril, I.H.; Zaqaibeh, B. Computer virus strategies and detection methods. *J. Open Probl. Comput. Sci. Math.* **2008**, *1*, 29–36.
70. Dube, T.; Raines, R.; Peterson, G.; Bauer, K.; Grimaila, M.; Rogers, S. Malware Target Recognition via Static Heuristics. *Comput. Secur.* **2012**, *31*, 137–147. [CrossRef]
71. Nissim, N.; Moskovitch, R.; Rokach, L.; Elovici, Y. Novel Active Learning Methods for Enhanced PC Malware Detection in Windows OS. *Expert Syst. Appl.* **2014**, *41*, 5843–5857. [CrossRef]
72. Lu, H.; Wang, X.; Zhao, B.; Wang, F.; Su, J. ENDMal: An Anti-Obfuscation and Collaborative Malware Detection System Using Syscall Sequences. *Math. Comput. Model.* **2013**, *58*, 1140–1154. [CrossRef]
73. Tabarzad, M.A.; Hamzeh, A. A Heuristic Local Community Detection Method (HLCD). *Appl. Intell.* **2017**, *46*, 62–78. [CrossRef]
74. Akhtar, M.S.; Feng, T. Malware Analysis and Detection Using Machine Learning Algorithms. *Symmetry* **2022**, *14*, 2304. [CrossRef]

75. Williams, C. Applications of Genetic Algorithms to Malware Detection and Creation. December 2009. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=32fe925452da3a44ba92dc3df051d0ccc9061980 (accessed on 6 July 2021).
76. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode Sequences as Representation of Executables for Data-Mining-Based Unknown Malware Detection. *Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
77. Yildiz, O.; Doğru, I.A. Permission-Based Android Malware Detection System Using Feature Selection with Genetic Algorithm. *Int. J. Softw. Eng. Knowl. Eng.* **2019**, *29*, 245–262. [CrossRef]
78. Wang, L.; Gao, Y.; Gao, S.; Yong, X. A New Feature Selection Method Based on a Self-Variant Genetic Algorithm Applied to Android Malware Detection. *Symmetry* **2021**, *13*, 1290. [CrossRef]
79. Sinha, S.; Bailey, M.; Jahanian, F. Shades of Grey: On the Effectiveness of Reputation-Based;Blacklists. In Proceedings of the 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE), Alexandria, VA, USA, 7–8 October 2008; pp. 57–64.
80. Choi, Y.H.; Han, B.J.; Bae, B.C.; Oh, H.G.; Sohn, K.W. Toward extracting malware features for classification using static and dynamic analysis. *Comput. Netw. Technol.* **2012**, 126–129.
81. Eskandari, M.; Khorshidpour, Z.; Hashemi, S. HDM-Analyser: A Hybrid Analysis Approach Based on Data Mining Techniques for Malware Detection. *J. Comput. Virol. Hack Tech.* **2013**, *9*, 77–93. [CrossRef]
82. Damodaran, A.; Troia, F.D.; Visaggio, C.A.; Austin, T.H.; Stamp, M. A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. *Comput. Virol. Hack Tech.* **2017**, *13*, 1–12. [CrossRef]
83. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A Survey of Intrusion Detection Techniques in Cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
84. Samsudin, K.; Al-baltah, I.A.; Al-Habshi, M.M. SCARECROW: Scalable Malware Reporting, Detection and Analysis. *J. Converg. Inf. Technol.* **2013**, *8*, 1–12.
85. Qiao, Y.; Yang, Y.; Ji, L.; He, J. Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences. In Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 16–18 July 2013; pp. 265–270.
86. Park, Y.; Reeves, D.S.; Stamp, M. Deriving Common Malware Behavior through Graph Clustering. *Comput. Secur.* **2013**, *39*, 419–430. [CrossRef]
87. Islam, R.; Tian, R.; Batten, L.M.; Versteeg, S. Classification of Malware Based on Integrated Static and Dynamic Features. *J. Netw. Comput. Appl.* **2013**, *36*, 646–656. [CrossRef]
88. Chen, Z.; Roussopoulos, M.; Liang, Z.; Zhang, Y.; Chen, Z.; Delis, A. Malware Characteristics and Threats on the Internet Ecosystem. *J. Syst. Softw.* **2012**, *85*, 1650–1672. [CrossRef]
89. Salem, M.B.; Hershkop, S.; Stolfo, S.J. A Survey of Insider Attack Detection Research. In *Insider Attack and Cyber Security*; Stolfo, S.J., Bellovin, S.M., Keromytis, A.D., Hershkop, S., Smith, S.W., Sinclair, S., Eds.; Advances in Information Security; Springer: Boston, MA, USA, 2008; Volume 39, pp. 69–90, ISBN 978-0-387-77321-6.
90. DeBarr, D.; Ramanathan, V.; Wechsler, H. Phishing Detection Using Traffic Behavior, Spectral Clustering, and Random Forests. In Proceedings of the 2013 IEEE International Conference on Intelligence and Security Informatics, Seattle, WA, USA, 4–7 June 2013; pp. 67–72.
91. Scheutz, M.; Andronache, V. Architectural Mechanisms for Dynamic Changes of Behavior Selection Strategies in Behavior-Based Systems. *IEEE Trans. Syst., Man Cybern. B* **2004**, *34*, 2377–2395. [CrossRef]
92. Lauf, A.P.; Peters, R.A.; Robinson, W.H. Embedded Intelligent Intrusion Detection: A Behavior-Based Approach. In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), Niagara Falls, ON, Canada; 2007; pp. 816–821.
93. Moon, D.; Im, H.; Kim, I.; Park, J.H. DTB-IDS: An Intrusion Detection System Based on Decision Tree Using Behavior Analysis for Preventing APT Attacks. *J. Supercomput.* **2017**, *73*, 2881–2895. [CrossRef]
94. Zhao, Y.; Bo, B.; Feng, Y.; Xu, C.; Yu, B. A Feature Extraction Method of Hybrid Gram for Malicious Behavior Based on Machine Learning. *Secur. Commun. Netw.* **2019**, *2019*, 1–8. [CrossRef]
95. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Differential Area Analysis for Ransomware Attack Detection within Mixed File Datasets. *Comput. Secur.* **2021**, *108*, 102377. [CrossRef]
96. Jung, S.; Won, Y. Ransomware Detection Method Based on Context-Aware Entropy Analysis. *Soft Comput.* **2018**, *22*, 6731–6740. [CrossRef]
97. McIntosh, T.; Jang-Jaccard, J.; Watters, P.; Susnjak, T. The inadequacy of entropy-based ransomware detection. In *Proceedings of the International Conference on Neural Information Processing, Sydney, Australia, 12–15 December 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 181–189.
98. Kang, M.; Won, J.; Park, J.; Kim, J. A CNN-Based Encrypted Data Detection for Ransomware Defense. *KIISE Trans. Comput. Pract.* **2022**, *25*, 279–283. [CrossRef]
99. Wojnowicz, M.; Chisholm, G.; Wolff, M.; Zhao, X. Wavelet Decomposition of Software Entropy Reveals Symptoms of Malicious Code. *J. Innov. Digit. Ecosyst.* **2016**, *3*, 130–140. [CrossRef]
100. Ko, J.; Kwak, J. Accuracy Enhancement of Determining File Encryption Status through Divided Shannon Entropy. In Proceedings of the Korea Information Processing Society Conference, Jeju, Republic of korea, 11 October 2018; pp. 279–281.

101. Lyda, R.; Hamrock, J. Using Entropy Analysis to Find Encrypted and Packed Malware. *IEEE Secur. Privacy Mag.* **2007**, *5*, 40–45. [CrossRef]
102. Sorokin, I. Comparing Files Using Structural Entropy. *J. Comput. Virol.* **2011**, *7*, 259–265. [CrossRef]
103. Baysa, D.; Low, R.M.; Stamp, M. Structural Entropy and Metamorphic Malware. *J. Comput. Virol. Hack Tech.* **2013**, *9*, 179–192. [CrossRef]
104. Dewan, P.; Kashyap, A.; Kumaraguru, P. Analyzing Social and Stylometric Features to Identify Spear Phishing Emails. In Proceedings of the 2014 APWG Symposium on Electronic Crime Research (eCrime), Birmingham, AL, USA, 23–25 September 2014; pp. 1–13.
105. Ndibanje, B.; Kim, K.; Kang, Y.; Kim, H.; Kim, T.; Lee, H. Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction. *Appl. Sci.* **2019**, *9*, 239. [CrossRef]
106. Trinius, P.; Holz, T.; Gobel, J.; Freiling, F.C. Visual Analysis of Malware Behavior Using Treemaps and Thread Graphs. In Proceedings of the 2009 6th International Workshop on Visualization for Cyber Security, Atlantic City, NJ, USA, 11 October 2009; pp. 33–38.
107. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware Images: Visualization and Automatic Classification. In Proceedings of the Proceedings of the 8th International Symposium on Visualization for Cyber Security-VizSec' 11, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
108. Ahmadi, M.; Ulyanov, D.; Semenov, S.; Trofimov, M.; Giacinto, G. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. In Proceedings of the Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9 March 2016; pp. 183–194.
109. Kancherla, K.; Mukkamala, S. Image Visualization Based Malware Detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16 April 2013; pp. 40–44.
110. Fredrikson, M.; Jha, S.; Christodorescu, M.; Sailer, R.; Yan, X. Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA; 2010; pp. 45–60.
111. Jiang, X.; Xu, D. Profiling Self-Propagating Worms via Behavioral Footprinting. In Proceedings of the 4th ACM workshop on Recurring, Alexandria, VA, USA; 2006; p. 17.
112. Jiankun, H. Host-Based Anomaly Intrusion Detection. In *Handbook of Information and Communication Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 235–255.
113. Ashoor, A.S.; Gore, S. Intrusion Detection System: Case study. In Proceedings of the International Conference on Advanced Material Engineering, Cairo, Egypt, 9 October 2011; Volume 15, pp. 6–9.
114. Murtaza, S.S.; Khreich, W.; Hamou-Lhadj, A.; Couture, M. A Host-Based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules. In Proceedings of the 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), Pasadena, CA, USA, 4–7 November 2013; pp. 431–440.
115. Kaur, H.; Gill, N. Host Based Anomaly Detection Using Fuzzy Genetic Approach (FGA). *IJCA* **2013**, *74*, 5–9. [CrossRef]
116. Cesare, S.; Xiang, Y. A Fast Flowgraph Based Classification System for Packed and Polymorphic Malware on the Endhost. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, Australia; 2010; pp. 721–728.
117. Song, D.; Brumley, D.; Yin, H.; Caballero, J.; Jager, I.; Kang, M.G.; Liang, Z.; Newsome, J.; Poosankam, P.; Saxena, P. BitBlaze: A New Approach to Computer Security via Binary Analysis. In *Information Systems Security*; Sekar, R., Pujari, A.K., Eds.; Lecture Notes in Computer Science; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 2008; Volume 5352, pp. 1–25.
118. Yoshioka, K.; Hosobuchi, Y.; Orii, T.; Matsumoto, T. Vulnerability in Public Malware Sandbox Analysis Systems. In Proceedings of the 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul, Republic of Korea, 19–23 July 2010; pp. 265–268.
119. Willems, C.; Holz, T.; Freiling, F. Toward automated dynamic malware analysis using cwsandbox. *IEEE Secur. Priv.* **2007**, *5*, 32–39. [CrossRef]
120. Inoue, D.; Yoshioka, K.; Eto, M.; Hoshizawa, Y.; Nakao, K. Automated malware analysis system and its sandbox for revealing malware's internal and external activities. *IEICE Trans. Inf. Syst.* **2009**, *92*, 945–954. [CrossRef]
121. Miwa, S.; MIYACHI, T.; ETO, M.; YOSHIZUMI, M.; SHINODA, Y. Design and Implementation of an Isolated Sandbox with Mimetic Internet Used to Analyze Malwares. In Proceeding of the 2007 USENIX Conference on Community Workshop on Cyber Security Experimentation and Test, Boston, MA, USA, 6–7 August 2007.
122. Lee, D. Analysis of Malware Detection Techniques Based on Machine Learning. Master's Thesis, Department of Convergence Service Security Engineering Graduate School of Soonchunhyang University, Asan, Republic of Korea, 2018.
123. Rathore, H.; Agarwal, S.; Sahay, S.K.; Sewak, M. Malware Detection Using Machine Learning and Deep Learning. In *Big Data Analytics*; Mondal, A., Gupta, H., Srivastava, J., Reddy, P., Somayajulu, D., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11297.
124. Liu, L.; Wang, B.; Yu, B.; Zhong, Q. Automatic Malware Classification and New Malware Detection Using Machine Learning. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 1336–1347. [CrossRef]
125. Bae, S.I.; Lee, G.B.; Im, E.G. Ransomware Detection Using Machine Learning Algorithms. *Concurr. Comput. Pr. Exper.* **2020**, *32*, e5422. [CrossRef]

126. Xu, Z.; Ray, S.; Subramanyan, P.; Malik, S. Malware Detection Using Machine Learning Based Analysis of Virtual Memory Access Patterns. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 169–174.
127. Singh, J.; Singh, J. Detection of Malicious Software by Analyzing the Behavioral Artifacts Using Machine Learning Algorithms. *Inf. Softw. Technol.* **2020**, *121*, 106273. [CrossRef]
128. Arslan, R.S.; Doğru, İ.A.; Barişçi, N. Permission-Based Malware Detection System for Android Using Machine Learning Techniques. *Int. J. Softw. Eng. Knowl. Eng.* **2019**, *29*, 43–61. [CrossRef]
129. Win, T.Y.; Tianfield, H.; Mair, Q. Big Data Based Security Analytics for Protecting Virtualized Infrastructures in Cloud Computing. *IEEE Trans. Big Data* **2018**, *4*, 11–25. [CrossRef]
130. Xingyuan, C.H.E.N.; Yuanzhao, G.A.O.; Huilin, T.A.N.G.; Xuehui, D.U. Research progress on big data security technology. *Sci. Sin. Inf.* **2020**, *50*, 25–66.
131. Choi, M.-J.; Bang, J.; Kim, J.; Kim, H.; Moon, Y.-S. All-in-One Framework for Detection, Unpacking, and Verification for Malware Analysis. *Secur. Commun. Netw.* **2019**, 1–16. [CrossRef]