Tuuli Veini

Detection techniques of common malware features: a systematic review

Master's Thesis in Information Technology

May 29, 2023

University of Jyväskylä

Faculty of Information Technology

Author: Tuuli Veini

Contact information: tuuliveini@gmail.com

Supervisor: Timo Hämäläinen

Title: Detection techniques of common malware features: a systematic review

Työn nimi: Tyypillisten haittaohjelmapiirteiden tunnistusmenetelmät: systemaattinen kirjallisuuskatsaus

Project: Master's Thesis

Study line: Information Technology

Page count: 77+9

Abstract: Building accurate and robust detectors is essential to keep up with constantly evolving malware. In this thesis, a systematic literature review of detection techniques of common malware features was conducted. Prevalent malware families of recent years were first studied to identify their common features, most important of which where API calls and communication with a Command and Control server. The systematic review was then conducted based on the discovered features. The final analysis included 33 papers published between 2018 and 2023. All reviewed papers applied behavior-based detection and most of them used machine learning in their proposed model. The papers suggested that building both accurate and fast detectors is possible with machine learning models, and feature processing techniques can be used to make detectors resistant to some evasive tactics used by malware. The study revealed a lack of research focus on optimizing the use of computational resources and counteracting sandbox evasion.

Keywords: malware, malware detection, antivirus, machine learning

Suomenkielinen tiivistelmä: Tarkkojen ja vakaiden haittaohjelmatunnistimien luominen on välttämätöntä haittaohjelmien kehittyessä jatkuvasti. Tässä pro gradu -tutkielmassa suoritettiin systemaattinen kirjallisuuskatsaus tyypillisten

haittaohjelmapiirteiden tunnistusmenetelmistä. Viime vuosien yleisimpiä haittaohjelmaperheitä tutkittiin ensin niille tyypillisten piirteiden tunnistamiseksi, joista tärkeimpiä olivat API-kutsut ja kommunikaatio komentopalvelimen kanssa. Sen jälkeen suoritettiin systemaattinen katsaus löydettyjen piirteiden perusteella. Analysoitavaksi valittiin 33 artikkelia, jotka oli julkaistu vuosien 2018 ja 2023 välillä. Kaikki käsitellyt artikkelit sovelsivat haittaohjelmien käyttäytymisen tunnistamista ja suurin osa käytti koneoppimista kehittämässään mallissa. Analyysin perusteella tarkkojen ja nopeiden tunnistimien kehittäminen on mahdollista koneoppimismalleilla, ja tunnistettavien piirteiden käsittelyllä voidaan torjua joitain haittaohjelmien käyttämiä väistötaktiikoita. Tutkimus osoitti puutteita laskentaresurssien käytön optimointiin ja analyysiympäristön välttämisen torjumiseen keskittyvässä tutkimuksessa.

Avainsanat: haittaohjelma, haittaohjelmien tunnistaminen, virustorjunta, koneoppiminen

List of Figures

List of Tables

Contents

1 Introduction

Malware is an ever increasing problem in the modern world. New malware is constantly being developed as cybercrime becomes a more popular, profitable, and sophisticated option for criminals. At the same time, thousands of people join the Internet every day, IoT devices are rapidly gaining popularity, and critical infrastructure is digitalized—all creating new opportunities for threat actors to target.

While new malware is constantly being developed, malware that originated years ago is also still going around. Agent Tesla is spyware that was originally discovered in 2014, and it was still one of the most often detected malware families in 2021—as was Qakbot, a Trojan used to steal banking credentials that was first found already in 2007 (CISA, ACSC August 2022). These malware families have developed over the years and several new versions of them have been released. Malware developers introduce new features and obfuscation techniques to the new variants which allows them to reuse the same type of malware without being detected. A recent example of this is a new Agent Tesla variant released in 2021 that incapacitated a Microsoft Windows antimalware interface and hid the malware's communications, which made both detecting and analyzing it more difficult (Threatpost February 2021).

In recent years, the COVID-19 pandemic has had a significant impact on increasing the number of cyber attacks. In a survey conducted in 2021, 74 % of US and UK financial institutions reported noticing a significant increase in cybercrime during the previous year, and many experienced that transitioning to remote work made them more vulnerable to cyber threats. Managing remote access programs, VPNs, and widely distributed employee devices (while also dealing with budget cuts) created new challenges for IT security teams and plenty of exploitation opportunities for criminals especially during the transition period. (Digital Intelligence 2021)

Another sector that suffered greatly from cybercrime during the pandemic was healthcare. At the end of 2020, cyber attacks against healthcare organizations

increased globally by 45 % over a two-month period, compared to an average increase of 22 % in other sectors. Attacks included Distributed Denial of Service (DDoS) attacks, remote code execution and botnets, but ransomware was by far the most prevalent attack method. Even before the pandemic hospitals have been a popular target for ransomware attacks because they are more likely to pay the ransom than victims in other sectors. Access to data is critical for hospitals to operate and ensure patient safety, and spikes in COVID-19 cases (like the one towards the end of 2020) created even more pressure on them to deal with attacks quickly. (Check Point January 2021)

A recent trend in the malware field is the increased popularity of the Malware-as-a-Service (MaaS) model. MaaS providers offer malware as one-time purchases, commissioned work, or even based on a subscription model. Outsourcing the technical work enables even less skilled cybercriminals to execute malware attacks. The services are especially of interest in the case of Ransomware-as-a-Service (RaaS) products since ransomware provides a straightforward revenue model for cybercriminals. The high demand causes RaaS providers to develop new ransomware in increasing amounts: in the first half of 2022 the amount of new ransomware almost doubled compared to the second half of 2021. (FortiGuard Labs 2022)

Existing detection methods have a hard time keeping up with the fast development of malicious software. Traditionally, antivirus programs rely on a signature-based detection where they compare potential threats against a database of known malware signatures. Thus they cannot detect new threats until they are identified with another method and updated in the database. Another common technique is behavior-based detection which monitors how potentially malicious software behaves in the system. It is better equipped to detect new types and variants of malware but heavier computational requirements prevent it from being used in everyday malware detection.

Recently machine learning has become a much more accessible tool for security researchers. Computational power has increased while its costs have decreased,

and machine learning as a field has progressed significantly. Additionally, labeled malware samples have become more readily available for researchers instead of only security professionals. (Gibert, Mateu, and Planes 2020) Machine learning is attractive as a potential malware detection method since it allows handling large masses of data, and it has the ability to generalize information and make decisions based on it—a quality that makes it powerful specifically in detecting new malware. While machine learning has become easier to use in research, the requirements for computational resources are still a challenge in applying it for general use.

This thesis gives an overview of the most common features used in malware in the recent years, and the detection techniques applied to them. The selected research method is the systematic literature review. The study is motivated by the lack of recent and comprehensive literature reviews on malware detection. Souri and Hosseini (2018) list several shortcomings in recent literature reviews, which include analyzing old articles and the lack of a systematic research method among other things.

The research questions are defined as follows:

Q1. What are the defining characteristics of the most common malware in active use in 2022?

Q2. What methods are best suitable for detecting said malware features?

The rest of this thesis is organized as follows. Chapter 2 describes the theoretical background of common malware types, detection techniques, previous literature reviews on malware detection methods, and features of recent malware. Chapter 3 describes the research design including background on conducting systematic reviews and a plan for how the research method is applied in this study. Chapter 4 presents the search results and a systematic review of the selected papers guided by the research questions. Finally, conclusions of the study are presented in chapter 5.

2 Theoretical background

In this chapter common malware types are introduced to provide background information and to highlight the range of different malicious programs that detectors need to take into account. Different malware detection approaches are then described along with limitations of current detectors. Concepts and challenges related to the application of machine learning on malware detection are described. Several surveys previously done on malware detection techniques are presented to provide additional motivation for the present study by showing that similar work has not been done in recent years. Finally, selected prevalent malware families are studied to identify their key features which are then described in detail to answer the first research question.

2.1 Malware classification

Malware types can be broadly divided into

- viruses,
- worms,
- spyware,
- adware,
- Trojans
- botnets,
- ransomware, and
- fileless malware.

The descriptions in this section are based on articles and definitions from Kaspersky Lab (n.d.), Malwarebytes (n.d.[a]), and F-Secure (n.d.[d]).

Viruses and worms are similar types of malware and worms can also be classified as a subtype of viruses. They can corrupt files, slow down the network connection and the system, spread to other devices, and they have the ability to self-replicate. There are three main differences between them. Firstly, viruses require a host file they are

attached to while worms are independent programs. Secondly, viruses require user interaction, such as running an executable or enabling macros on an infected file, before they are able to start replicating. Worms can self-replicate independently of user action which allows them to spread rapidly once the first instance is installed. Thirdly, viruses spread by replicating their code to other files. Worms, however, exist as a single instance in a system, and replicate themselves to other systems.

Spyware is software that is installed on the victim's device without their knowledge for the purpose of gathering information about the user. Typically it also slows down the performance of the infected system. It is usually installed along with seemingly legitimate software that the victim intentionally installs. The information gathered can be for example login credentials or credit card information, which the attacker can then use for identity theft or account takeover, or sell to third parties.

Adware plays advertisements automatically on the victim's device, often quite aggressively. It is typically installed as a part of free software and its purpose is to create revenue to the creator through playing ads. Another way to get adware is as a drive-by download by visiting an infected website where the adware then exploits a vulnerability in the browser to install the adware on the user's device. Once installed, it can play ads regardless of what websites the user visits or what browser they use.

Trojans are malware that are masked as legitimate files or software. Attackers may use phishing techniques to pass the Trojan to the victim for example as an email attachment, or the victim may install it as a part of free software. Their behavior depends on the payload—Trojans can contain e.g. a backdoor that gives attackers access to the system, a rootkit to hide their other malicious activity, or a keylogger to record the user's keystrokes.

A botnet is a network of remotely controlled devices. After being infected with malware the infected device, or bot, waits for instructions from the attacker either directly or through other bots. The owner of the device doesn't usually notice being infected. Botnets are often used for DDoS attacks to flood the target host with repeated requests to disrupt services and prevent legitimate requests from going

through. Other possible applications include e.g. mining cryptocurrencies and phishing campaigns. Botnet hosts may also rent them for other cybercriminals to use.

Ransomware is malware that encrypts files on the target device and may also lock the device completely. The attackers then demand a ransom in exchange for a decryption key that returns the files to the victim's use. While ransomware is a distinct type of malware, its infection method is based on some other malware type. For example, CryptoLocker was a Trojan that was distributed using a botnet (F-Secure, n.d.[a]), and WannaCry was a cryptoworm that self-propagated to other systems (Malwarebytes, n.d.[b]).

Fileless malware is a newer technique used for malware attacks. Fileless malware does not exist as an executable file in the device's file system but instead it is loaded directly into the device's Random Access Memory (RAM) which makes it difficult to detect with signature-based methods. It also uses legitimate utilities and libraries of the infected system (e.g. PowerShell, the .NET framework, and the Windows Task Scheduler) for its own malicious purposes which makes its behavior similar to a benign program. The data in RAM is cleared when the system is shut down which should stop the attack, but attackers can maintain persistence even after a reboot by e.g. setting up scripts that run when the machine is restarted. The clearing of RAM also makes learning about an attack after the fact difficult but some signs can be uncovered through memory forensics (Kaspersky Lab 2017).

As can be seen from the examples of ransomware, these malware categories are not absolute or mutually exclusive and they often overlap. Typically another type of malware is wrapped in a Trojan—for example, a Trojan can connect the infected computer to a botnet or install spyware on the device. The described types are also not the only forms of malware—some other types include e.g. logic bombs, and the aforementioned rootkits.

2.2 Detection techniques

Malware detection approaches are typically divided into two categories, signature-based and behavior-based detection. Both techniques can also be applied at the same time for a hybrid approach.

2.2.1 Signature-based detection

Most antivirus software is based on signature-based detection. Antivirus programs compare potential threats to known malware signatures and block the ones with a match in the program's database. The advantages of signature-based detection are its fast identification, ease of use, and accessibility, which make it an effective method against known threats.

Signature-based detection can be implemented in a few different ways. The simplest approach is to use the hashed value of a malicious file. It is a precise way to identify a specific malware and results in virtually no false positives. However, hash-based searching is especially ineffective against modifications in the malware as even the addition of one character to a text file alters its hashed value. (Griffin et al. 2009) A more advanced method to form a signature is by reverse engineering malware binaries. Performing static analysis on disassembled malware binaries can provide even more information about the malware's specific features that can then be used for the signature, like specific instructions used in the malware code (Deng and Mirkovic 2018). Static analysis is traditionally done manually but in recent years machine learning methods have been applied to automate the analysis process making it considerably faster (Hassen, Carvalho, and Chan 2017; Shalaginov et al. 2018). The application of machine learning on malware detection is covered more extensively in section 2.3.

Signatures can be improved by using a combination of features discovered in static analysis and other common characteristics like file size and type. A popular tool to create signatures is YARA[1], an open-source project that can be used to create

1. https://virustotal.github.io/yara/

descriptions based on textual or binary patterns in malicious files. By using a combination of rules the descriptions can match a whole family of malware which enables catching several malware variants with one signature.

The major drawback of signature-based scanners is that they can be evaded relatively easily through code obfuscation, which is a general approach to masking the malware's purpose by making the code more difficult for a human reader to understand. Code obfuscation can be done utilizing different techniques like encryption, polymorphism, dead code insertion, instruction substitution, and register reassignment (You and Yim 2010). Out of these, encryption is perhaps the most challenging to counteract as it prevents disassembling malware binaries for static analysis (Chen, Brophy, and Ward 2021). Encrypted malware consists of an encrypted virus body and a decryptor which decrypts the virus when it is run in the host system. Antivirus products cannot detect signatures in the encrypted virus body but they can still look for signatures in the decryptor. To counter this, decryptors can be improved further through polymorphism. Polymorphic malware can modify its decryptor using different obfuscation techniques and thus create countless different decryptors. Metamorphic malware, on the other hand, is not encrypted but can similarly mutate the virus body as it propagates, potentially making every instance of the malware unique. (Szőr 2005; You and Yim 2010).

Regardless of how the signature is formed, the reliance on previously identified malware remains the biggest disadvantage of the signature-based approach. It is virtually ineffective against new malware strains or new variants of older malware that have been obfuscated using some of the previously described methods. Creating a general signature for an entire malware family may be effective in detecting new variants, but it still requires prior knowledge and analysis of the malware before the signature can be formed and thus doesn't protect against the constantly emerging new malware. The concept is easily applied to antivirus programs but signatures need to be more robust than the traditionally used hashes and binary sequences. Signature-based detection is also, by principle, ineffective against fileless malware since it lies solely in memory and does not have a file that can be checked for a

signature. The heavy focus on malware behavior and machine learning methods in recent research is further indication that signature-based detection in itself is no longer sufficient for effective malware detection.

2.2.2 Behavior-based detection

Behavior-based detection methods detect malware based on its activity when it infects a system. It is especially useful in detecting new malware variants because typically malware behavior remains the same even if malware authors attempt to avoid detection through code obfuscation. Even across different families, malware instances perform similar actions in the infected host which makes the behavior-based method effective against new threats. (Mosli et al. 2017; Singh and Singh 2021)

Typically behavior-based tools utilize dynamic analysis by running a potentially malicious program in a sandbox environment (e.g. Cuckoo[2]) and logging its behavior in a text file. Activity monitored in the sandbox includes typically API calls and their arguments, created processes, accessed domains and IP addresses, file or registry access patterns, loaded DLLs, and DNS requests (Mosli et al. 2017; Maniriho, Mahmood, and Chowdhury 2022b). In addition to the Windows API, services provided by the Windows kernel can be accessed with system calls. Using the services in itself does not necessarily indicate malicious behavior but calling them directly in user code is rare and often a sign of malicious activity (Raff and Nicholas 2020). Sandbox programs can also provide a quick static analysis of the malware code (Fox May 2021). The log file is usually analyzed manually and the information can be used to create a signature for the malware (David and Netanyahu 2015).

Malicious activity can be divided into host behavior and network behavior. Host behavior refers to activities that may alter the attributes of the host system, like modifying registry files (Lin et al. 2015). Network behavior typically means activities like connecting to remote hosts or servers to exfiltrate data or send spam (Perdisci, Lee, and Feamster 2010; Lin et al. 2015). Monitoring network-level activity may

2. `https://cuckoosandbox.org/`

increase the detection accuracy because in some cases, malware variants that execute different actions in the system show the same malicious behavior in the network (Perdisci, Lee, and Feamster 2010). This is especially useful when considering new variants of a malware family that try to avoid detection by altering their behavior in the system. However, the inverse is also possible. Malware authors may maintain the same system-level code but modify network attributes—for example, malware that originally used the IRC protocol to contact a Command and Control (C&C) server is modified to use HTTP instead (Perdisci, Lee, and Feamster 2010). This way the two malware samples behave very similarly in the host but differently in the network, and both behavior types need to be considered for ideal accuracy.

The main downside of the behavior-based approach is that it is much slower compared to signature-based solutions as it requires running and analyzing files in a sandbox. Additionally, some malware can detect it is running in a sandbox environment and alter its behavior to seem like a benign file or stop executing completely. For example, the fileless ransomware UIWIX terminates itself if it detects a VM or a sandbox (Trend Micro May 2017). When malware passes detection and is allowed to run in the host, the malicious behavior is activated. The sandbox should thus be set up in a way that malware cannot identify the analysis environment based on e.g. a standard MAC address or lack of user files and cache. (Singh and Singh 2021)

Despite its limitations, the behavior-based approach is often considered much more powerful—and even necessary—compared to the signature-based method which is insufficient in the constantly evolving field of malware. Behavior-based techniques are immune to code obfuscation because they focus on what the malware is doing rather than how it is implemented, which makes them much more accurate for detecting both known and unknown threats. (Moser, Kruegel, and Kirda 2007; O'Kane, Sezer, and McLaughlin 2011) The increasing popularity of new types of malware like fileless malware also highlights the need for dynamic analysis as it may not be possible to detect them with signatures (Saad, Briguglio, and Elmiligi 2019).

2.3 Machine learning in malware detection

Machine learning (ML) has gathered a lot of interest as an option to overcome the downsides of traditional malware detection methods. This is aided by the recent advancements in machine learning like natural language processing and computer vision, the increase in computational power and the decrease in its cost, and the increase in publicly available labeled feeds of malware (Gibert, Mateu, and Planes 2020). ML methods allow handling large masses of data and finding patterns in them that would normally require a lot of manual work to uncover. The ability of ML models to generalize based on previously learned information is also an important advantage in detecting new malware variants (Raff and Nicholas 2020).

Applying ML to malware detection follows the same principles as the traditional methods. Patterns that ML solutions detect can be static features like binary sequences or other signatures, or dynamic features like memory and registry usage, network traffic, or API call traces (El Merabet and Hajraoui 2019; Gibert, Mateu, and Planes 2020). For example, Hassen, Carvalho, and Chan (2017) extracted different features using static analysis and used them to classify malware samples into known malware families, while David and Netanyahu (2015) applied dynamic analysis for an automated signature generation method based on malware's behavior and used it to train a deep neural network. As previously stated, static features can easily be altered or masked with different obfuscation techniques, which makes the dynamic features more useful when considering detecting new malware also in the case of ML methods.

Clustering and classification are two pattern identification methods commonly used in the automatic analysis of malware. Clustering is a form of unsupervised learning that identifies similarities between objects in unlabeled data and groups them according to their common characteristics. Traditional manual labeling of new malware samples is time-consuming work that could be eased with the use of clustering. (Ng et al. 2019) Classification, on the other hand, is a supervised learning technique to assign labeled objects into predefined groups. A malware classifier is trained using labeled datasets to distinguish between benign and malicious files,

and give accurate predictions when facing previously unseen files (El Merabet and Hajraoui 2019). Each file in the training dataset is labeled according to its defining features and whether it is malicious or benign, which enables learning algorithms to map the relationships between file features and labels (Markel and Bilzor 2014). Both clustering and classification applications utilize the fact that malware variants of the same family often share common static features and behavior patterns.

In conventional ML techniques feature engineering—the process of selectively extracting features from raw data—is done manually which requires extensive knowledge of the domain in addition to being time-consuming and error-prone (Maniriho, Mahmood, and Chowdhury 2022b). In recent years, deep learning (DL) solutions have contributed to great advancements in fields like speech recognition and natural language processing. DL algorithms have a hierarchical structure that imitates the human brain and allows automatically extracting features and abstractions from underlying data. Thus the algorithms have the ability to learn from both labeled and unlabeled datasets and produce classification models which automates the feature engineering process (Najafabadi et al. 2015; Maniriho, Mahmood, and Chowdhury 2022b).

There are still various challenges related to the application of ML in practical everyday malware detection. While more labeled datasets have become publicly available, finding a balanced, good-quality dataset is still a challenge. Training a model on a dataset with noticeably different sizes of classes can easily skew the model's predictions. Malware families with strong polymorphism are typically overrepresented in datasets that are filtered by sample-uniqueness and thus cause class imbalance (Rossow et al. 2012). Additional imbalance is caused by the ratio of malicious to benign samples. Virus samples are shared publicly but obtaining non-copyright-protected benign files is more difficult (Gibert, Mateu, and Planes 2020). In classifier training, researchers often use their own datasets or build on top of existing ones because many of the available datasets are outdated or limited due to challenges with privacy policies (Vinayakumar et al. 2019).

Using ML does not completely eliminate the challenge that evolving malware

creates. One issue related to this is concept drift—relationships between input and output change over time because malware, like any software, changes with added features, bug fixes, and porting to new environments (Gibert, Mateu, and Planes 2020; Raff and Nicholas 2020). ML models still need to be updated to keep up with the changes. Dynamic analysis techniques are an advantage against updating malware, but just as in traditional behavior-based detection, ML solutions that apply dynamic analysis are also ineffective against malware that is able to detect it is running in a virtual environment (Mohaisen, Alrawi, and Mohaisen 2015).

Another challenge related to ML is the interpretability of models. Most ML models used today are black box models that are given an input and they produce an output through a process that is invisible to humans. However, in malware detection applications it is more useful to have interpretable models to help analysts understand the logic behind the model's predictions and to discover if a model's decisions are biased or inconsistent (Maniriho, Mahmood, and Chowdhury 2022a). It can be especially useful when dealing with false positive cases to help analysts figure out whether the false alarm was due to an error in the training set or the model. The more interpretable a model is, the easier it is to assess its quality and correct it when needed. (Gibert, Mateu, and Planes 2020; Maniriho, Mahmood, and Chowdhury 2022b)

Despite the development of faster and more powerful processors, computational cost is still a major limitation of machine learning applications. Developing a practical and effective ML model is always a compromise between costs and accuracy. Several features need to be considered for most accurate classification but the more features are included in a classifier, the more computational resources it requires. Features need to be selected deliberately to mitigate the costs and take full advantage of the available resources. The development of more powerful processors could allow using more complex ML models in the future.

2.4 Related work

Numerous review papers on malware detection have been published over the years but malware develops fast and makes such reviews quickly outdated. The goal of the comparisons in this section is to show that this thesis does not overlap with previously published review papers. For relevancy's sake, only surveys from the last few years are covered in this section.

As Maniriho, Mahmood, and Chowdhury (2022a) mention in their survey, conducting multiple surveys on a similar topic is common and while they share the topic, they differ on scope, analysis, reviewed articles, and outcome. No survey can completely exhaust a topic. Additionally, the reviewed articles may be selected differently between otherwise similar surveys, whether it is the time frame, inclusion and exclusion criteria, or the article selection method. For example, Maniriho, Mahmood, and Chowdhury (2022a) used the snowball approach where they chose an initial batch of papers and used their bibliographies for selecting additional articles. In this thesis, however, the papers are selected based on the relevancy algorithms of the used search engines. The search and selection process is covered in more detail in section 3.3.

In a recent survey, Aboaoja et al. (2022) review malware analysis and detection approaches. They present a taxonomy that maps analysis and detection methods to the malware features most frequently associated with them, and divides feature engineering phases to different feature extraction and representation methods. While the research topic is very similar, they cover articles published over a much longer time compared to this thesis and they do not focus or comment on the most prevalent malware of the recent years.

Souri and Hosseini (2018) survey data mining techniques used in malware detection. They summarize the current challenges related to data mining, review current ML approaches, and cover the most significant techniques in detail. They compare the frequency and accuracy of classification methods used in the reviewed articles but do not specify the considered malware features.

Maniriho, Mahmood, and Chowdhury (2022a) review dynamic malware analysis tools and behavior-based detection techniques involving ML and DL techniques. For selecting the reviewed papers, they select a starting set of papers and expand the selection with additional papers from the references of the initial set, and with papers citing the initial group. The main difference compared to this thesis, in addition to the selection method, is that Maniriho, Mahmood, and Chowdhury (2022a) do not specify the most common features in recent malware. Additionally, in this review the scope is not limited to the behavior-based method as the goal is to present the techniques recently applied for malware detection regardless of the approach.

In another review, Tayyab et al. (2022) examine statistical, ML, and DL techniques in malware detection and review datasets and performance metrics used in past studies. Their specific focus in DL is the use of smaller datasets in meta learning algorithms like Few Shot Learning (FSL). The idea of FSL is to concentrate on the most significant features for best efficiency which could be useful when designing a cost-effective malware detector. However, they cover articles over a longer time period rather than the most novel research which is distinctly different from the purpose of this thesis.

There are numerous other surveys that focus specifically on ML techniques. El Merabet and Hajraoui (2019), Gibert, Mateu, and Planes (2020), Raff and Nicholas (2020), and Singh and Singh (2021) all examine the application of ML techniques on malware detection but they differ from this thesis in various ways. They may not state the features that are considered in detection, or they include older articles in their review because their goal is to paint an overall picture on the topic rather than describe the most recent features and methods. Conversely, some surveys, like Moussaileb et al. (2021), have similar research questions to this thesis but they limit their scope only to ransomware.

This systematic review examines the common features present in malware in the recent years, and the most efficient techniques to detect those features. As gathered in this section, there has been extensive work on reviewing malware detection but in most cases the surveys focus on the application of a specific detection method,

e.g. deep learning algorithms. Many reviews limit their scope to a general type of malware like ransomware instead of considering what malware features are most prevalent today. Additionally, the majority of the papers related to this topic are surveys describing the topic overall, rather than systematic reviews assessing the current state-of-the-art. Therefore in most of the summarized papers, the search protocol used for the reviewed articles is not described and cannot be directly compared with this review.

2.5 Features of recent malware

The first research question of this thesis is, Q1. *What are the defining characteristics of the most common malware in active use in 2022?* Examining features of the recently most widespread malware families helps identify commonalities between them. The information can be used to determine the key features that should be the focus of detection solutions to make them as efficient as possible.

Check Point lists Emotet, Formbook, XMRig, AgentTesla, Trickbot and Ramnit frequently among the top malware families in their 2022 monthly reports. In January 2022, all of them were in the top 10 malware families (Check Point February 2022), and in December 2022 all except Trickbot made the list (Check Point January 2023). On the other hand, Kaspersky (December 2022) list Ramnit, ZeuS, and Trickbot as some of the top banking Trojans detected most often in 2022, and Magniber as one of the top ransomware families. The eight selected malware families and their commonly used techniques are reviewed in this section.

ZeuS, or Zbot, is a large family of Trojans that is mainly used to steal online banking information. It can intercept data entered in online forms, inject additional fields to the targeted websites, or even present a whole fake website hosted on a server controlled by the attacker. ZeuS communicates with remote servers to download its configuration file, send the stolen information, and download copies of itself or configuration files if needed. Typically it creates a hidden folder where it places files used to store the stolen information and the initial configuration file it downloads. It

also makes a copy of itself and if it doesn't detect any firewall processes, it modifies registry entries to establish automatic execution at startup. (F-Secure, n.d.[b]; Trend Micro June 2015) For example, if the variant drops the executable `sdra64.exe`, it adds it to the following registry entry (Trend Micro June 2015):

```
HKEY_LOCAL_MACHINE\Software\Microsoft\
  Windows NT\CurrentVersion\Winlogon
userinit = "%System%\userinit.exe, %System%\sdra64.exe,"
```

Ramnit, also known as Nimnul and Cosmu, is a worm with several variants developed since its creation. Ramnit originally spread through removable drives and infected files in the system but was soon updated with code copied from the ZeuS malware which enabled it to be used as a banking Trojan to steal user credentials (Dunn August 2011). Ramnit registers itself as a system service by adding registry keys to the `HKLM\SYSTEM\CurrentControlSet\Services\` registry tree which enables its automatic execution at system startup, and creates additional registry keys (similar to ZeuS's) to ensure further persistence (Trend Micro September 2014). For evasion purposes, it can add several registry values to hide itself from Windows Defender (Praszmo September 2017), delete registry values related to the Windows safe mode (Trend Micro September 2014), and kill processes associated with antivirus software (Chen November 2012).

Trickbot is a Trojan mainly used to steal financial data. It is primarily spread through malicious Microsoft Office macros which, once enabled, execute a script that downloads the main executable that contains the malware logic. As a persistence technique it creates a scheduled system task to keep itself running. (Salinas and Holguín June 2017) It has several modules with different functions, allowing it to e.g. steal browser data, inject malicious code into web browsers to monitor online banking details, gather network information in the infected system and send it to a C&C server, and steal credentials from different applications—among many other features. Because of its modular structure, it can be used for various kinds of attacks in addition to simply stealing banking data. (Ang May 2019)

Magniber is a ransomware program that can exploit various different vulnerabilities to gain access to a system where it encrypts files matching to predefined extensions (Trend Micro Research January 2023). Its encryption process has two phases: first it encrypts the files with a symmetric encryption key, then encrypts the key with a stronger asymmetric encryption algorithm. It modifies registry values to elevate its privileges and then runs a command to delete all shadow copies so that the encrypted files cannot be recovered. (Cybereason September 2021). During execution, it uses direct system calls rather than the standard Windows API which helps it evade detection. Recently variants have disguised themselves as Windows updates where the inital dropper is a JavaScript file which loads the .NET executable directly in memory, making the malware itself fileless. (Manna-Browne December 2022; Schläpfer October 2022)

Emotet started out as a banking Trojan and has expanded into a botnet. The Trojan infects machines typically through malicious Microsoft Office macros which download the executable malware from a remote server. Emotet has modules for stealing banking details from network traffic, stealing email credentials from email software, harvesting browser data, and extracting email addresses for spamming. The collected information is sent to a C&C server. (Symantec July 2018) It uses packing and code jumps for obfuscating the malware code (Trend Micro Research, n.d.), as well as various obfuscation techniques for the initial dropper (Silverio et al. May 2022). It can also inject code to `explorer.exe` and other processes which is a technique used for defense evasion and privilege escalation. For persistence, Emotet adds the executable malware payload to the Windows autorun registries, or creates scheduled tasks to execute it. Additionally it creates files in the system root directories to run them as services in an attempt to spread the malware to adjacent systems. (CISA January 2020) Since its evolution into a botnet, Emotet has been used to distribute other malware and recently it has been spreading the banking Trojan IcedID and the cryptominer XMRig (Proofpoint November 2022).

Formbook is a spyware Trojan used to steal login credentials from browsers and other applications. In addition to harvesting login details, it has various other

capabilities like logging keystrokes, searching through files, and taking screenshots. Before execution it checks the environment for debuggers and other analysis tools and terminates if they are found. With specifics depending on the Formbook variant, it drops various script files to `%APPDATA%\Local\Temp\` and adds the main script files to a `Run` registry key to automatically execute them. Formbook then injects itself into the legitimate Windows processes `AddInProcess32.exe` and `explorer.exe` to mask its behavior. From then on it injects itself into various other processes like browsers and messaging applications from which it steals input and clipboard data. It can receive additional downloads and commands from a C&C server which it also contacts to send the stolen information. (F-Secure, n.d.[c]; Zhang April 2021a)

XMRig is an open-source cryptocurrency miner that, when used for malicious purposes, mines the cryptocurrency Monero on the victim's machine using their computational resources and electricity. It is not designed to be malicious in itself but attackers can deploy it to a breached system, or bundle it with other malware to mine cryptocurrency alongside the main attack (Check Point, n.d.). For example, the Sysrv botnet spreads the miner by targeting vulnerabilities that allow remote code access (Kimayong April 2021). Running a miner in the background makes general use of the computer much slower but detecting the miner itself can still be difficult. The processes XMRig runs have been concealed with obfuscation techniques like process hollowing (Remillano II, Nebre, and Dela Cruz December 2019), or by hiding the process names with a separate tool (Masubuchi May 2021). The specific indicators of compromise and viable detection techniques are thus dependent on the malware it is paired with and other attack techniques used.

Agent Tesla is a Remote Access Trojan (RAT) and a keylogger used to steal login credentials from various browsers, email clients, and other software, as well as other information gathered through harvesting clipboard data and capturing screenshots (Zhang June 2017). It sends all collected information to a C&C server over FTP, HTTP, or SMTP. The initial script is often executed through malicious Microsoft Office macros but other types of files have also been used to distribute the malware.

It establishes persistence through registry keys or scheduled tasks, or by copying itself into the Windows Startup folder, and uses various process injection methods to hide itself. (Walter April 2021; Splunk Threat Research Team November 2022) Some indicators that can be used to detect it include e.g. excessive use of `taskkill.exe`, creation of scheduled tasks, modification of registry keys for privilege escalation, or suspicious processes started by Microsoft Office macros (Splunk Threat Research Team November 2022).

Based on the review in this section the key techniques used by the recent major malware families are modifying registry entries, scheduling tasks, process injection, creating processes, and contacting a C&C server. The techniques are commonly related to establishing persistence in the infected system, e.g. by ensuring that the malware runs at specific time intervals or at system startup. The techniques and how different malware use them are next described in more detail.

Windows has the following registry keys to define applications that are run automatically every time a user logs on (Microsoft Learn February 2022):

```
HKEY_LOCAL_MACHINE\Software\Microsoft\
  Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\
  Windows\CurrentVersion\Run
```

The `Winlogon` key used in Windows 7 systems has the following two subkeys that can autostart programs (MITRE ATT&CK June 2022a):

```
HKEY_LOCAL_MACHINE\Software\Microsoft\
  Windows NT\CurrentVersion\Winlogon\Shell
HKEY_LOCAL_MACHINE\Software\Microsoft\
  Windows NT\CurrentVersion\Winlogon\Userinit
```

The `userinit` key is used by e.g. ZeuS (Trend Micro June 2015). Additionally, the `Load` value of the following registry keys can be used to run programs when a user logs on (MITRE ATT&CK June 2022a):

```
HKEY_LOCAL_MACHINE\Software\Microsoft\
  Windows NT\CurrentVersion\Windows Load
HKEY_CURRENT_USER\Software\Microsoft\
  Windows NT\CurrentVersion\Windows Load
```

For example Agent Tesla saves its executable to the `Load` value (Zhang June 2017). Registry key manipulation can be detected by monitoring command execution, file modification, process creation, and Windows registry key creation and modification (MITRE ATT&CK June 2022a).

Task scheduling can be used for the initial execution of the malware, or as a persistence technique to keep it running. All major operating systems have utilities for scheduling programs or scripts to run at a specific time. In Windows systems the Task Scheduler is responsible for scheduling tasks. In addition to the GUI, it can be accessed with e.g. the `schtasks` or `at` utilities. (MITRE ATT&CK July 2022e) For example, Trickbot schedules two tasks: one that runs at system startup, and one that executes every minute. Typically the scheduled executables are located in `%APPDATA%\Roaming\` which helps distinguish Trickbot's scheduled tasks from benign system tasks. (Campbell and Cargill July 2020) Agent Tesla uses `schtasks` to execute the malware, and one of its variants attempts to download a new version of it every two hours to keep the malware up-to-date (Zhang December 2021b). Emotet is also known to use scheduled tasks for persistence (CISA January 2020).

Process injection is a technique used to avoid detection by masking the malware process under a legitimate process. It can also be used for privilege escalation since executing code in the context of another process may allow inheriting the memory, system and network resources, and privileges it holds. Process injection can be done by creating a new thread within the target process, or by hijacking one of its existing threads, and is commonly done by abusing legitimate functionalities. (MITRE ATT&CK October 2022d; Secarma, n.d.) Agent Tesla uses process injection by executing a legitimate program like `RegAsm.exe` or `Regsvcs.exe` and injecting its malicious code into the started process. It commonly uses process hollowing, a technique where memory sections of a legitimate process are unmapped and

replaced with malicious code. (Walter April 2021)

Creating system processes, or services as they are called in Windows systems, is another technique for achieving persistence. The services run on system startup and are generally used to perform background system functions. Services can be created using system utilities like `sc.exe`, by modifying the registry, or via the Windows API. (MITRE ATT&CK June 2022b) For example, Trickbot creates an autostart service that runs the malware on system startup (Llimos and Pascual November 2018), and Emotet creates files in root directories that are run as services to propagate the malware to other systems (CISA January 2020).

In addition to different system activities, most—if not all—malware today perform some network activity as well. Many of the malware families reviewed above contact a C&C server to download malicious scripts or configuration information, and to send the data they have collected. The contacted domains and IP addresses vary between malware (and often between variants or even individual instances of the same variant) so they can be used as signatures only for specific cases. Detecting malicious network activity without a known signature or domain is very challenging, and the analysis of traffic can be made more difficult with encryption, like in the case of TrickBot (Salinas and Holguín June 2017). Applying network features for general malware detection is thus very difficult.

Lastly, it is essential to consider the API calls malware makes during its execution. The way malware performs many of its actions like creating processes is often done through API calls (MITRE ATT&CK April 2022c). For example, Formbook calls various Windows APIs like `ZwOpenProcess()` and `ZwOpenThread()` to inject itself into `explorer.exe` (Zhang April 2021a). XMRig calls the `CreateProcess()` API to start `AddInProcess.exe`, and then calls several other APIs like `VirtualAlloc()` and `WriteProcessMemory()` to deploy the malicious `xmrig.exe` in that process (Zhang January 2023). Accordingly, API calls are one of the most used features in behavior-based detection (Gibert, Mateu, and Planes 2020; Singh and Singh 2021).

3 Research design

The goal of this study is to identify detection techniques that are best suitable for the common features present in the most prevalent recent malware. The selected research method is the systematic literature review. This section describes the general approach of a systematic review as well as the research questions, the search strategy, the selection criteria, and the data extraction plan for the systematic review conducted in this thesis.

3.1 Systematic literature review

As Kitchenham and Charters (2007) define it, a systematic literature review aims to identify, evaluate, and interpret all available research relevant to a particular research question or topic. Whereas primary studies collect data directly, a systematic review is a form of secondary study that uses primary studies as its data source. It is a method used to summarize existing evidence regarding a topic, identify gaps in current research, and provide a base for further research (Kitchenham and Charters 2007). It is thus a suitable method to obtain a comprehensive overview on the topic of interest.

Compared to systematic mapping studies, systematic reviews have more narrowly defined research questions and scope. Systematic mapping studies aim to give an overview of the types of studies published on a particular research area. Systematic reviews, on the other hand, study a more specific topic and evaluate the selected papers in detail. Systematic reviews require more depth and effort to achieve their goals and therefore cover fewer articles than mapping studies. (Petersen et al. 2008)

Due to the nature of a Master's thesis, the study selection and data extraction processes are carried out by a single author which limits the available resources considerably. It also diverges from the guidelines of Kitchenham and Charters (2007) as the lack of another perspective may cause biased results, but it is accepted as a risk. The limited resources, as well as the time constrains that come with a

Master's thesis, are taken into account in the number of papers selected for analysis.

In the process detailed by Kitchenham and Charters (2007), conducting a systematic review can be divided into three main phases: planning, conducting, and reporting the review. Planning involves identifying the need for a review, specifying the research questions, and defining the review protocol. Conducting the review consists of searching and selecting primary studies, data extraction, and data synthesis. Finally the results are written in a report. The following sections describe how each phase is implemented in this thesis.

3.2 Research questions

A systematic review is driven by the research questions as they direct the outcome of each phase of the review process. The search and selection processes identify primary studies that address the research questions. The data extraction process extracts information related to the research questions from the selected primary studies. Finally the data analysis process synthesizes the extracted data to form answers to the research questions. (Kitchenham and Charters 2007)

A research question should be meaningful to both practitioners and researchers, lead to changes in current practices or increased confidence in the current practice, and identify common beliefs that conflict with reality (Kitchenham and Charters 2007). In the context of malware detection, it is in the interest of both researchers and industry security professionals to develop detection solutions that are as accurate and efficient as possible, though for industry practitioners the performance aspect may be more critical. Identifying the best techniques for detecting common malware features informs whether efforts should be directed to building new kinds of solutions or improving the most critical aspects of the existing ones.

Staples and Niazi (2007) emphasize the importance of defining the research questions narrowly to reduce the intense workload of a systematic review and to improve the quality of the paper selection and data extraction. Following this advice, this thesis is limited to two research questions with a very limited scope.

Based on the guidelines described above, the research questions are defined as follows:

Q1. What are the defining characteristics of the most common malware in active use in 2022?

Q2. What methods are best suitable for detecting said malware features?

Identifying the common features of the most prevalent recent malware families is a prerequisite for answering the second question since it allows narrowing the search strings used during the search process. This additional objective is presented as the first research question but it is not included as a part of the formal systematic review because the information is for the most part gathered from gray literature, including various electronic sources like blog posts and white papers published by commercial security providers, which makes defining and implementing a systematic search plan challenging if not impossible. The question is answered in the background portion of this study in section 2.5 where eight malware families that were among the most widespread ones in 2022 are described along with their features.

The second research question is the main question answered with a systematic analysis of what detection techniques were used in academic literature. Information gathered from answering the first question is used to formulate the search strings and direct the data extraction process.

3.3 Search and selection process

The resources used to search for articles were IEEE Xplore[1], Google Scholar[2], ACM Digital Library[3], and ScienceDirect[4]. Brereton et al. (2007) note that a single source is not enough to find all primary studies on a subject. Several databases were used also in this thesis to make the search as complete as possible within reasonable bounds.

The most relevant keywords related to the research topic were identified as *malware*

1. https://ieeexplore.ieee.org/Xplore/home.jsp
2. https://scholar.google.com/
3. https://dl.acm.org/
4. https://www.sciencedirect.com/

and *detection*. Initial trial searches revealed that using those search terms alone gave quite general results, so the malware features reviewed in section 2.5 were incorporated into the search string. The top results of the trial searches also included many Android and Internet of Things (IoT) specific articles, but the focus of this thesis is limited primarily to the Windows platform. To refine the search closer to the scope of the present study, papers that clearly focused on mobile or IoT malware were excluded from the search. To achieve this, papers that contained the terms *android*, *mobile*, *iot*, or *internet of things* in their title were excluded. Surveys and reviews were also excluded from the search since the papers needed to be primary studies. Boolean conditions were used to include all relevant search terms into the same string. Based on these conditions, the following search string was formulated:

```
malware AND detection AND
  (api OR registry OR task schedule OR process creation OR
  process injection OR "command and control") NOT
  title:android NOT title:mobile NOT title:iot NOT
  title:"internet of things" NOT
  title:survey NOT title:review
```

A trial search on Google Scholar using the above rules and time frame limited to 2018 onward gave about 17 300 results. This showed more promising papers in the top results compared to the other trial searches and was selected as the base search string used in the systematic review.

The same base search string is used for each search engine and its specific syntax is modified according to that of the used search engine. The results are sorted according to relevance based on each search engine's own algorithm, and the time frame is limited to 2018–2023 using the search engine's filter function. The 50 first results are initially retrieved from each search. Duplicate papers are then removed and the inclusion and exclusion criteria (detailed in the next section) are applied to choose the most relevant papers. The selection is based on a review of title, keywords, abstracts, and conclusions as per the recommendation by Brereton et al. (2007). If necessary, additional selection rounds are conducted with more

scrutiny to narrow down the selected papers.

3.4 Inclusion and exclusion criteria

In addition to well-defined search strings, it is necessary to define selection criteria to limit the reviewed material to the most relevant primary studies that provide evidence about the research question (Kitchenham and Charters 2007).

For a paper to be selected, it was required for the paper to clearly refer to applying a technique to malware detection or classification in the abstract or the conclusions. Classification is treated synonymously to detection because especially in ML applications, many studies emphasize classification over detection but both principles can be applied for detection solutions. Similarly, trial search results included some papers that described their focus as malware analysis rather than detection. The topics can overlap depending on how the author interprets and discusses them, and the full extent to which they are covered may not be apparent based on abstract and conclusions alone. However, papers that focus on malware analysis but make no mention of a detection technique are discarded to make the data extraction process less taxing, even at the risk of discarding some papers that discuss detection outside of the abstract and the conclusions. Additionally, the paper had to state that the proposed technique was applied to at least one of the key features identified in section 2.5.

The included papers were required to be primary studies as the systematic review is a secondary study. It was also required for them to be peer-reviewed journal articles or conference papers. This was especially important because rigorous quality assessment was not done in the scope of this thesis due to time limitations. The quality of the publishing channels was futher assessed based on the evaluation by Publication Forum (JUFO)[5]. The journal or conference of each paper was searched in the JUFO Portal[6]. Conferences not found in the search were assessed based on their

5. https://julkaisufoorumi.fi/en
6. https://jfp.csc.fi/en/web/haku/julkaisukanavahaku

publisher according to JUFO's classification policy (Publication Forum September 2016). Requirements for level 1 are e.g. transparency on the editorial board and the peer-review process, and if a publishing channel fails to fulfill any of the the requirements it is classified as level 0 (Publication Forum March 2023). Therefore, papers that did not reach at least level 1 in the JUFO evaluation were excluded.

The other exclusion criteria were related to practical matters and further ensuring the relevancy of the papers. Papers that were not written in English were excluded to reflect the language of this thesis. Due to practical limitations, papers not accessible for free or with the University of Jyväskylä credentials were also excluded. To narrow down the results to match the scope of this review, papers that focused specifically on mobile or IoT malware, or malware of some other non-Windows platform, were excluded.

Finally to further ensure relevancy, papers published before 2018 were excluded. Even though only the top malware of 2022 was used to review the most common malware features, the search for the detection techniques was expanded to about a five-year period. This was due to two main reasons. First, the malware that was the most popular in 2022 was not developed in that year alone, but has been around for years. Second, the features are not unique to the reviewed malware but have been used in a similar manner for years, and thus have been studied by security researchers over a long period of time. The search is still limited to relatively recent publications to focus on techniques that are still relevant in the malware field today.

To summarize, the inclusion criteria were:

- The topic is malware detection or classification
- Primary study
- Peer-reviewed journal article or conference paper
- Targets one or more of the following features:
  - API calls
  - Modifying registry entries
  - Scheduling tasks

  - Process injection
  - Creating processes
  - C&C communication

The exclusion criteria were:

- Published before 2018
- Language is not English
- Full text is not available for free or with University of Jyväskylä credentials
- Focuses on mobile, IoT, or some other non-Windows platform
- Publishing channel evaluated as level 0 on Publication Forum

3.5 Quality assessment

Kitchenham and Charters (2007) recommend assessing the quality of the selected articles based on bias, internal validity, and external validity. Bias refers to how much the results systematically depart from the 'true' results. Internal validity refers to the degree of systematic error likely produced in the study. External validity defines how largely the effects are applicable outside of the study. Staples and Niazi (2007) express that it is difficult for an outsider to reliably assess how thoroughly other authors have considered the validity of their studies, and use a simplified quality assessment approach of extracting a 'yes' or 'no' based on whether the study had considered each quality factor. They discovered that only 15 % of their selected studies addressed the quality measures even partially.

Brereton et al. (2007) report previously using an approach where they evaluated the quality of the papers based on the completeness of the data. Papers lacking in quality did not provide data or statistical tests to support their claims, or reported some but not all of the required information. In another of their previous studies, Brereton et al. (2007) report relying on the quality of the peer-review processes of the journals their selected papers were published in.

In this thesis, rigorous quality assessment is not done for the selected papers because of the time limitations discussed earlier. Instead, a simplified version is adapted

from the guidelines of Kitchenham and Charters (2007). Additionally, even though Kitchenham and Charters (2007) recommend performing the quality assessment before data extraction, in this thesis the quality criteria are considered as part of the extracted data to simplify the process.

For this thesis it is relevant to consider the accuracy of the detection techniques. Accuracy is here defined as the correctly classified instances divided by all instances. It can be quantified in more detail with precision, recall, and the F-score. Precision is the number of true positive results divided by the number of all positive results (including false positives), and recall is the number of true positives divided by the number of all results that should have been labeled as positive. Together they are used to calculate the F-score. As an additional quality indicator, the transparency of the research process from the experiment setup to the conclusions is considered.

The following checklist is used for quality assessment:

1. Are precision and recall considered when evaluating the accuracy of the method?
2. Is clear evidence provided to support the claims made in the study?

3.6 Data extraction

The purpose of planning the data extraction stage is to accurately record the information to be obtained from the reviewed papers. Defining a data extraction form directs the analysis process to gather all information relevant to the research questions. (Kitchenham and Charters 2007)

According to Biolchini et al. (2005), the extracted information can be objective or subjective. Objective information is the identification details of the study (title, authors, and source), methodology used to conduct the study, results obtained through executing the study, and limitations found by the article's authors. Subjective information is additional information obtained directly from the authors, or conclusions the reviewer makes after reading the study.

The data extraction questions are formed with the research questions in mind to obtain all relevant data without too much excess information. General publication information including the title, author names, publication year, and publishing journal or conference is gathered from all analyzed papers.

The following questions are used for data extraction:

1. What is the studied detection technique(s)?
2. What malware feature(s) is the detection technique applied to?
3. Is the detection technique based on static or dynamic features?
4. How are the features extracted?
5. Is the model based on non-ML or ML techniques?
6. How accurate is the technique?
7. What is the computational performance of the technique like?
8. What are the advantages of the technique?
9. What are the disadvantages of the technique?

In addition, answers to the quality assessment questions defined in the previous section are extracted.

3.7 Data synthesis

In data synthesis, the results of the primary studies are summarized and collected together. Data synthesis can be descriptive, quantitative, or qualitative depending on the type of analyzed studies and the research questions (Kitchenham and Charters 2007).

In descriptive synthesis the extracted data is collected into a table that includes information relevant to the research questions, and highlights the similarities and differences between study results. It is important to identify whether the results are consistent between studies or if they differ, and what causes the differences. (Kitchenham and Charters 2007) Simply tabulating the information may not clearly answer the research questions, so it is necessary to explain inductively how the collated information relates to the questions (Brereton et al. 2007).

This thesis applies primarily descriptive synthesis by tabulating the key information extracted from the papers and examining it in relation to the research questions. As a quantitative measurement, the accuracy of the models proposed in the papers is included in the table to aid in reliably and clearly comparing the different detection techniques. As observed by Brereton et al. (2007), comparing quantitative information may be difficult as the reporting protocols can vary considerably between studies, which is why the accuracy is accepted as a sufficient quantitative measure.

4 Literature review of detection techniques of common malware features

In this section the results of the systematic literature review are presented. The goal of this section is to answer the second research question, Q2. *What methods are best suitable for detecting* [*the most common*] *malware features?*

4.1 Search results

The time frame in all searches was limited to 2018–2023 according to the selection criteria. The searches were conducted in March and April of 2023.

The search string used for Google Scholar was:

```
malware AND detection AND
  (api OR registry OR task schedule OR process creation OR
  process injection OR "command and control")
  -intitle:android -intitle:mobile -intitle:iot
  -intitle:"internet of things"
  -intitle:survey -intitle:review
```

For ACM, the same search string gave more irrelevant results so the search string was modified slightly for the rest of the searches. The search string used for ACM was:

```
malware AND detection AND
  (api OR registry OR task schedule OR process OR inject OR
  "command and control") NOT
  [Title: android] NOT [Title: mobile] NOT
  [Title: iot] NOT [Title: "internet of things"] NOT
  [Title: survey] NOT [Title: review]
```

For IEEE Xplore, a similar approach was applied and the search string was:

```
malware AND detection AND
 (api OR registry OR task schedule OR process OR
 inject OR "command and control") NOT
 "Document Title":android NOT
 "Document Title":mobile NOT "Document Title":iot NOT
 "Document Title":"internet of things" NOT
 "Document Title":survey NOT "Document Title":review
```

For ScienceDirect, the main search string was:

```
malware AND detection AND
  (api OR registry OR task schedule OR process OR
  inject OR "command and control")
```

As an additional filter, in the "Title" field of the advanced search it was specified to exclude irrelevant search terms:

```
NOT android NOT mobile NOT iot NOT "internet of things" NOT
  survey NOT review
```

For each search, the first 50 results were initially selected. Three selection rounds were performed to select the papers for review. During the first selection, papers were evaluated roughly for their relevancy. Instead of considering specific malware features in this phase, the papers only needed to be related to malware detection or classification to complete the first selection more quickly. All other selection criteria detailed in section 3.4 were also considered except the publishing channel's JUFO level. After the first selection round and removal of duplicates, 135 papers were selected.

During the second selection round the topic of the papers was evaluated with more scrutiny based on the malware features of interest. If the abstract or the conclusions did not specify them, the full article was skimmed for the considered features. This was especially necessary in the case of ML solutions because the models are often trained with multiple features and the authors may not describe the feature selection in the abstract and conclusions. After accepting only papers with relevant features,

79 papers were selected.

Finally on the third selection round, papers were selected based on their publisher's quality rating as evaluated by JUFO. Five papers were discarded because their publishing channels were classified as level 0. After the selection rounds and quality assessment, a total of 74 papers were selected for analysis. The results of these selection rounds are summarized in table 1.

Table 1: Summary of the search and selection rounds.

Search engine	**Search results**	**Potential papers**	**First accepted**	**Discarding duplicates**	**Accepted based on malware features**	**Accepted based on JUFO level**
Google Scholar	18100	50	26	26	21	17
IEEE Xplore	779	50	43	39	26	25
ACM	3070	50	33	33	11	15
ScienceDirect	2940	50	38	37	21	27
All	24889	200	140	135	79	**74**

4.2 Analyzed articles

The number of papers was still very large for a single researcher to analyze. To narrow down the sample articles, half of the papers were randomly selected with a Python script. The list of articles was first randomized using the `shuffle()` function in the `random` module, then the first 37 papers in the randomized list were selected and the rest discarded.

During the analysis, three more papers were discarded because they covered a large number of features that were unclear. Additionally, one paper was discarded because it was found to be about an analysis tool rather than a detection solution. In the end, a total of 33 papers were analyzed. The final selection rounds are summarized in table 2. The full list of articles used for final analysis is presented in appendix A. The discarded papers are listed in appendix B.

The analyzed papers were published between 2018 and 2023 as per the selection criteria. They were distributed across the publication years with five papers published in 2018, six papers in 2019, 11 papers in 2020, three papers in 2021, seven papers in 2022, and one in 2023.

Table 2: Summary of the randomized articles selected for analysis.

Search engine	**After selection rounds**	**Half of the papers randomized**	**Final analysis**
Google Scholar	17	9	7
IEEE Xplore	25	11	11
ACM	11	9	8
ScienceDirect	21	8	7
All	74	37	**33**

4.3 Quality assessment

The quality of the papers was assessed with a short checklist during data extraction. The quality questions were:

1. Are precision and recall considered when evaluating the accuracy of the method?
2. Is clear evidence provided to support the claims made in the study?

Most papers assessed the accuracy of their method by calculating precision, recall, and/or the F-score. Some papers calculated the true positive rate (TPR), which is the same as recall, and the false positive rate (FPR) instead. In total, 22 papers used both precision and accuracy, six papers used TPR and FPR, four papers provided only the accuracy or used different additional metrics, and one paper showed various values for accuracy only in graph form.

For the most part the analyzed papers fulfilled the second quality standard but there were a few instances where some of the research steps were unclear or difficult to understand. Typically the feature extraction process was not well described. For example, Goyal and Kumar (2020) extracted string features with static analysis but did not expand on the features or the extraction process, or whether the extracted features were processed somehow before classifier training. In hindsight, the phrasing of the quality question leaves room for interpretation which is why quantities for papers that did or did not fulfill it are not stated here.

4.4 Malware features

The majority of the papers studied malicious API calls as a primary feature. Papers included both API calls monitored dynamically and static API calls extracted from PE files. The second-most-represented feature was C&C communication and only one paper was focused on registry activity. The rest of the papers studied a variety of features including those identified in section 2.5, as well as features like file operations, PE file headers, mutex operations, and using a packer. The distribution of the papers across different features is represented in figure 1. In cases where there were several features covered in equal manner in the paper and it could not be clearly classified under one of the main features, the paper was classified under a variety of features.

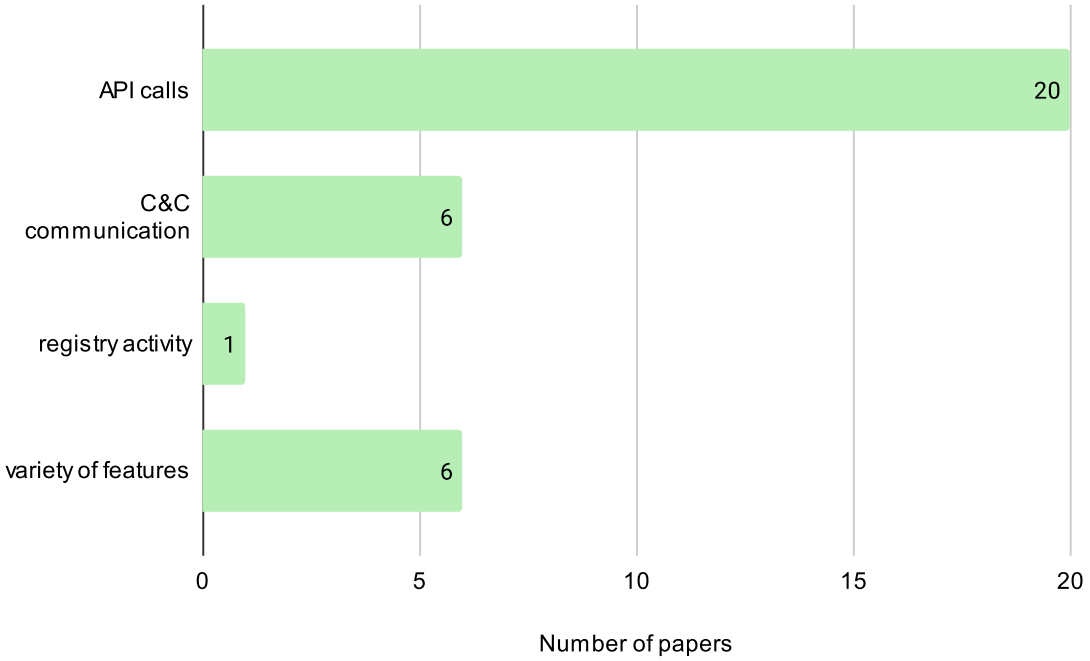

Figure 1: Number of analyzed papers for each malware feature.

Task scheduling was identified as key behavior in many malware families in section 2.5 but none of the analyzed papers identified it as a targeted behavior for detection. Instead, it is likely that if it is relevant for certain malware, it is detected as part of the various API calls characteristic of that malware. Including it as a key behavior to analyze was perhaps unnecessary due to inexperience and lack of understanding on my part. In similar fashion, registry activity can be detected based on API calls which may explain why there was only one paper focusing on registry operations in the analysis pool. Additionally, process creation and injection appeared in papers alongside other features but none of the papers focused solely on either of them.

The techniques used to detect API calls and their accuracies are summarized in table 3 and those used for C&C detection in table 4. The paper on detecting registry activity is shown in table 5 and the remaining papers on various features are summarized in table 6.

Table 3: API detection techniques and their accuracies.

Paper	**Detection technique**	**Detection accuracy**
Yin et al.	HIN	98.25%
He and Kim	CNN, ResNet-50	shown in graph
Liu and Wang	BLSTM	97.85%
Lu et al.	combined RF and bidirectional residual LSTM model	96.70%
Zhang et al.	CNN (opcode) and BPNN (API)	F1 95%
Goyal and Kumar	kNN, GNB, MNB, DT, SVM, RF	97.53% (RF on API)
Kishore, Barisal and Mohapatra	CNN with LeNet-5, AlexNet and ResNet-34; double detector model with system call and API detectors	99.40%
Kishore, Barisal and Mohapatra	double detector model with SVM and HTM detectors	95.2%
Li, Lü and Shi	GAN	TP 98.4% (original), 98.7% (retrained)
Namani and Khan	ANN	94.62% (API), 97.42% (with PE headers)
Rabadi and Teo	SVM, GB, RF, DT, PA	99.2–99.4% (GB)
Wu, Guo and Wang	GB	99.97%
Amer, Zelinka and El-Sappagh	MLE	99.7% (Windows dataset)
D'Angelo, Ficco and Palmieri	Associative Rule	99.03%
Ding et al.	BERT combined with GCN	97.82%
Li et al.	GNN	98.43%
Namita, Prachi and Sharma	ML models trained with DT, SVM, LR and kNN	99.91% (SVM and LR)
Wijaya, Lim and Kotualubun	Markov chain	87.19%
Xu, Zhang and Zhou	BERT	88%
Xue, Wang and Feng	MLP	91%

Table 4: C&C communication detection techniques and their accuracies.

Paper	**Features**	**Detection technique**	**Detection accuracy**
Ghafir et al.	IP addresses, SSL certificates, domains, Tor connections	separate detection module for each feature, correlation framework finds links between alerts	TP 82.3% FP 13.6%
Piet, Anderson and McGrew	HTTP, TLS, custom plaintext protocols, polling behavior, and tool-specific artifacts	passive scanning and semi-active scans targeting suspicious servers triggered by passive monitoring system, RF classifier	TP 98.5% FP 0.01%
Chen et al.	statistical behavioral features from packets, network flow converted to grayscale images	double detector model with One-Class SVM anomaly detector (statistical behavior profiling) and Convolutional Auto-encoder anomaly detector (spatial behavior profiling)	F1 94.1%
Alageel and Maffeis	domain names requested in PCAP files	parser module, crawler module, preprocessor module, and classifier module; RF classifier performed best	98.51%
Novo and Morla	encrypted malware C&C traffic, list of TLS/SSL certificates used by botnet C&C servers, extract numerical features from traffic flows	3-layer DNN detector trained on a public C&C traffic dataset, white-box adversarial learning, and a proxy-based approach for crafting longer flows	95% (original), varies with adversarial samples
Spooren et al.	domain generation algorithms	1) classical ML using manually engineered features (RF) 2) DL recurrent neural network (LSTM)	93.8% (RF), 98.7% (LSTM)

Table 5: Registry activity detection technique and its accuracy.

Paper	**Detection technique**	**Detection accuracy**
Tajoddin and Abadi	anomaly-based ensemble classifier with Gaussian, kNN, k-means and Parzen window classifiers	98.43%

Table 6: Detection techniques for combinations of features and their accuracies.

Paper	**Features**	**Detection technique**	**Detection accuracy**
Íncer Romeo et al.	registry modifications, processes, digital signing, packer detection, static and dynamic imports, file operations	monotonic GB classifier	temporal detection rate 62%
Pektaş and Acarman	mutexes, created processes, copying or deleting itself, DNS requests, remote IPs, TCP and UDP ports, reading files, registry keys, changed and created files	online learning algorithm	98%
Almutairi et al.	C&C communication: network's flow records; host behavior: file creation, registry key modification, processes	hybrid technique combining network and host data analyzers, classifiers with NB and DT algorithms	99.6% (hybrid+DT)
Gupta and Rani	file metadata, size, packer detection, sections, DLLs, dropped files, API calls, mutex operations, and file, registry, network and process activities	ensemble learning model with five base classifiers: NB, kNN, DT, SVM, RF	99.5% (method 1), 99.2% (method 2)
Ravi et al.	API calls, PE headers, PE imports, PE image	Attention-based Multi-View DL architecture using 1-d CNN, LSTM, and DNN	98% (Windows dataset)
García, DeCastro-García and Castañeda	API calls, network traffic, file activity, registry activity	homogeneous transfer learning algorithms with integrated ML algorithms	MCC 0.9531 (GB average)

4.5 Detection techniques

Most of the papers applied machine learning methods and a minority studied non-ML techniques. The ML models proposed in the papers used classical machine learning as well as neural networks and deep learning techniques.

Commonly used ML models were Random Forest (RF), k-Nearest Neighbor (kNN), Decision Tree (DT), Support Vector Machine (SVM), and Gradient Boosting (GB). RF models performed generally well. For example, out of all models studied by Goyal and Kumar (2020), RF was the best performing one with an accuracy of 97.53 %. Gupta and Rani (2020) built an ensemble learning model with kNN, DT, SVM and RF as some of the base classifiers and reached an accuracy of over 99 % with two different methods where one was based on weighted voting and one on stacking a set of base classifiers.

The GB applications were all implemented using the XGBoost library. The models by Rabadi and Teo (2020) and Wu, Guo, and Wang (2020) reached a very high accuracy of over 99 %. GB performed well also when integrated into transfer learning algorithms (García, DeCastro-García, and Castañeda 2023). However, the monotonic GB classifier proposed by Íncer Romeo et al. (2018) performed very poorly with a temporal detection rate of 62 %. The monotonicity was applied as a defense against evasive techniques used by malware but it caused a significant drop in accuracy when compared to the ordinary XGBoost classifier.

Google's language model BERT (Devlin et al. 2018) was used in two papers. Ding et al. (2022) applied it to extract semantic information from API sequence data and their model reached a high accuracy of 97.82 %. Xu, Zhang, and Zhou (2022) added API calls to BERT's pretraining data to improve its classification accuracy in malware detection applications but the accuracy remained quite low at 88 %.

Other ML classifiers used were Naive Bayes (NB), Gaussian Naive Bayes (GNB), Multinomial Naive Bayes (MNB), Hierarchical Temporal Memory (HTM), Passive-Aggressive (PA), Logistic Regression (LR), Gaussian, k-means, and Parzen window. Some of the other ML techniques used were the framework Heterogeneous

Information Network (HIN), and a new ML algorithm based on associative rules proposed by D'Angelo, Ficco, and Palmieri (2021).

Neural networks were implemented most often implemented with Convolutional Neural Network (CNN), Graph Neural Network (GNN), Artificial Neural Network (ANN), Deep Neural Network (DNN), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BLSTM). All the techniques performed well with high accuracies e.g. in the CNN-based double detector model by Kishore, Barisal, and Mohapatra (2020b), the GNN-based detection and classification framework by Li et al. (2022), and the DL architecture applying CNN, LSTM and DNN by Ravi et al. (2022) which all reached over 98 % in accuracy.

Other neural network techniques used were Back Propagation Neural Network (BPNN) and Graph Convolutional Networks (GCN). Multilayer Perceptron (MLP), Generative Adversarial Network (GAN), and Convolutional Auto-encoder were three types of ANNs used in the papers.

Only a few papers used non-ML techniques and they varied depending on the detected features. C&C communication is typically detected by scanning the network for suspicious traffic. In the framework proposed by Ghafir et al. (2018) the malicious traffic was for the most part detected based on lists of known IP addresses, SSL certificates and Tor servers. Amer, Zelinka, and El-Sappagh (2021) proposed a model based on the statistical Maximum Likelihood Estimation (MLE) method that reached a very high accuracy of 99.7 % in detecting malware based on API calls. In another paper, Wijaya, Lim, and Kotualubun (2022) applied a stochastic Markov chain on API call categorization but their model had poorer performance at just over 87 % in accuracy.

4.6 Detection techniques of common malware features

This section identifies what detection techniques are typically applied to API calls, C&C communication, registry activity, and other malware features. Additionally it discusses commonly used feature extraction and processing techniques.

4.6.1 API calls

Across studies API calls were usually extracted by running the malware samples in the Cuckoo sandbox which executes the files and logs the API calls made by the sample. Alternatively, in some papers the API calls were extracted statically from PE headers or from disassembled binaries. Raw API data was typically filtered for nonrepetitive consecutive calls to condense it into the most relevant information.

In a few papers, API calls were examined in conjunction with opcode (operations performed in machine language instructions) or system calls resulting in a high detection accuracy. Zhang et al. (2019) used BPNN to train their model for detecting API calls and CNN for opcode. The feature-hybrid model was more accurate than either method alone and retained similar detection speed. Kishore, Barisal, and Mohapatra (2020a) built a double detector model where the SVM model first detected samples based on system calls and then sent the samples initially classified as benign to a HTM model that re-classified the files based on API calls.

A common technique for feature processing was to represent API sequences in graph form. Using APIs called by malware to form an API call graph helps retain more characteristic information than simple call names (Li, Lü, and Shi 2020). The GAN model used by Li, Lü, and Shi (2020) required a fixed size for the input samples so they identified the key APIs—those that reflect the characteristics of malware—to build graphs of the same scale. In the Associative Rule based model by D'Angelo, Ficco, and Palmieri (2021), all transitions between two API calls were extracted and, in conjunction with the order of their occurrence, were represented in graph form. Ding et al. (2022) connected the API sequences of multiple processes into a graph that included information about API sequences called by a process, as well as parent-child relationships between processes. Wijaya, Lim, and Kotualubun (2022) formed API call category transition probability graphs from call sequences and used them to identify malware based on similarity scores.

To improve accuracy, some papers suggested including the arguments used with the API calls, or the processes related to them. Considering API arguments makes

the feature extraction process more complex but the behavior and implications of API calls can differ significantly based on the arguments—for example, calling the `WriteFile` API function on system files can have very different impact compared to writing on user files (Li et al. 2022). Rabadi and Teo (2020) incorporated the arguments with two different methods: in the first they extracted all arguments of each API call as one feature, and in the second extracted each argument of each API call as a separate feature. Both methods performed well with the second method applied to a GB model providing the best detection accuracy. Li et al. (2022) argue that simply extracting the arguments, like was done by Rabadi and Teo (2020), is not sufficient. In the model they propose, they included semantic information like relationships between API calls in addition to the calls and their arguments.

Two of the papers applied converting the malware binaries to grayscale or RGB images and detecting the API call patterns in the images. Kishore, Barisal, and Mohapatra (2020b) built a double detector CNN model with one detector for API calls and one for system calls and achieved a high detection accuracy with the LeNet-5 architecture. Their model performed well with a detection speed of 1.03 s which is slow in comparison to the state-of-the-art models but could be improved with reducing noise in the malware images. He and Kim (2019) also built a CNN model using the ResNet-50 model but their solution performed siginificantly worse. CNNs generally need to have a fixed size input which in the case of malware samples means cutting the binaries to the same size and potentially losing some of the malware code. He and Kim (2019) applied Spatial Pyramic Pooling (SPP) to allow inputs of arbitrary size but it resulted in very high false positive rates.

It is clear that retaining more details related to the API calls, rather than using only the API names, is beneficial for the detector's performance. API call graphs and grayscale images are both viable ways to represent the more complex information. An alternative way to increase the detector's accuracy is to combine API call detection with system calls or opcode. High detection accuracies can be achieved with both traditional ML and DL models, though the former methods typically have much more practical computational requirements.

4.6.2 C&C communication

The studies on C&C communication varied based on the techniques and the detected features. The features used by different studies were numerous: IP addresses, TLS/SSL certificates, domain fluxing, Tor connections, HTTP packets, polling, domain name requests, and so on. Most of the papers based their detectors on generic behavior that could be applied to a variety of malware. In some cases the detection was limited to static signatures or features of specific malware families.

A simple way to build a network detector is to block known addresses, domain names, and certificates used by malicious actors. Ghafir et al. (2018) proposed a model of four submodules that detect malicious IP addresses, malicious SSL certificates, domain fluxing, and Tor connections. The domain fluxing module monitored the network behavior but the three other modules were based on denylists of known C&C server IPs, malicious SSL certificates, and Tor servers. Static lists have a similar problem as signature-based detection in the sense that they can make the detectors quickly out of date. To mitigate this issue, Ghafir et al. (2018) implemented automatic update mechanisms to refresh the denylists while the detector is running. Relying on known addresses and certificates is still a disadvantage as it makes the detector slow to react to new malware.

Ghafir et al. (2018) based their fourth module on detecting domain generation algorithms (DGAs) which was also studied by Spooren et al. (2019). Detecting DGAs is an alternative method to relying on lists of domains known to be malicious. DGAs are used in domain fluxing to change the C&C server's domain name frequently in order to avoid detection by denylists. Ghafir et al. (2018) based their detection on a case where an infected host queries a large number of domains in an attempt to connect to the C&C server. If a large number of failed queries was detected while monitoring the network, the detector raised an alert. Spooren et al. (2019) built RF and LSTM solutions to detect DGA names themselves. They applied a dataset based on manually crafted features, like character sequences, as well as other features, like whether the DGA domain has a valid top level domain, on the RF model. Their second model was based on LSTM that can learn the patterns of

DGAs from raw input data. They reached high accuracy when detecting individual DGAs but multiclass detection had poor performance on both models.

Domain names were also studied by Alageel and Maffeis (2021). They focused on detecting Advanced Persistent Threats (APTs) which are threat actors that target specific organizations. Their model was based on different features characteristic of APT domains, and they found that DGA domains were not commonly used by APTs. They built separate modules for parsing DNS queries from PCAP files, crawling identifying features from them, preprocessing and normalizing the data, and finally classifying it. They reached a high accuracy of over 90 % but since their solution is based on discovering very specific types of threats, it is not sufficient alone for practical detection solutions.

Piet, Anderson, and McGrew (2018) implemented passive and active network scanning for detecting three commonly used RATs: Meterpreter, Empire, and Pupy. Their passive scanner monitored the network for generic behavior of the attack tools, and the active scanning was used when the passive scanner failed to identify a suspicious server. Their solution can be used to detect the three malware families studied reliably but as such it cannot be applied more broadly because the behavior is specific to those families. For example, each attack tool had unique characteristics in their HTTP packages that were used to identify them. On the other hand, polling behavior, i.e. querying the C&C server for new commands, is more generic across different RATs and could thus be utilized also in common detectors. The active scanning was done manually to target uncertain cases which made it resource and time consuming, and should not be applied to all traffic because of its invasive nature. In the end, the principle of detecting generic behavior is much more reliable against new variants since it does not rely on static signatures.

Chen et al. (2020) used an anomaly-based approach where they trained one detector with normal statistical features from packets, and one detector with normal network flow data converted to grayscale images. The statistical detector was based on a One-Class SVM model and the image detector on a Convolutional Auto-encoder ANN. A high false positive rate is typical of anomaly detection models which is why only the

network flows labeled as abnormal by both detectors were regarded as C&C traffic. They reached precision and recall of over 90 % when the threshold to classify network flow as malicious was set relatively low.

Novo and Morla (2020) proposed a model for detecting encrypted C&C traffic that aims to be resistant against adversarial samples, i.e. data that attempts to deceive ML classifiers. They built a DNN detector that originally reached an accuracy of 95 % and then tested it with iterative hardening and attacking with crafted adversarial samples. Based on their results, the competition between adversarial attacks and retraining the model seems difficult to predict and defend, as an attack with new adversarial traffic can drop the detector's accuracy significantly.

As can be seen from the reviewed papers, network threats vary widely which can make it difficult to build a single generic solution to detect them all. An ensemble-based approach with multiple specialized modules based on different types of C&C communication may be an effective way to alleviate this issue.

4.6.3 Registry activity

The only paper solely on registry-based detection was a study by Tajoddin and Abadi (2019). Similarly to the C&C detector by Chen et al. (2020), they used an anomaly-based approach where they used only benign registry behavior to train an ensemble classifier. They ran benign programs in the Cuckoo sandbox and extracted all registry operations performed by them and used them to form feature vectors for training the classifier. They used Gaussian, kNN, KM, and Parzen to build a total of 40 subclassifiers, and used a subset of them for detection reaching an accuracy of over 98 %.

An ensemble classifier is a viable method to attain a high accuracy and reduce misclassifications but using multiple classifiers increases the complexity significantly. Tajoddin and Abadi (2019) used a memetic algorithm to reduce the ensemble classifier's size and achieve a log-linear complexity $\mathcal{O}(r \log r)$ for the detection step. However, their solution had multiple classifiers for each method

which may be excessive, and from a resource perspective it could be more beneficial to have fewer carefully crafted submodules in the first place.

As mentioned in section 4.4, registries can be accessed with API calls. Windows provides API functions such as `RegCreateKeyExA` to perform read, write, and delete operations on registry keys and values. Solutions based on detecting registry activity are thus more of a minor subset of API-based detectors. Detectors may in fact benefit from considering calls related to multiple different malicious activities to improve detection accuracy.

4.6.4 Combinations of features

The remaining six papers included in the analysis used a variety of features in addition to the ones that were central to the literature search. The papers included features like mutexes (locking execution threads), creating files, and using a packer, as well as static features like file metadata and PE headers. All the papers used ML and DL techniques which make it easier to handle a large number of features in one detector.

API calls, C&C communication, and registry activity were typically detected based on similar principles as in the papers focusing primarily on those features. Almutairi et al. (2020) used an alternative method to detect registry activity: their model checked periodically if the registry keys had changed. The approach is reasonable for their model which aims to detect botnets in an early infection phase but for detectors that run samples in a sandbox, inspecting API calls and registry keys themselves is likely sufficient and more straightforward. On the other hand, actively checking registries may help catch malware that has bypassed detection by e.g. terminating itself in the sandbox environment.

Process activities were identified as a key feature in section 2.5 but the literature search did not result in any papers targeting specifically that behavior. However, four papers included it in their feature selection along with a variety of other features. Íncer Romeo et al. (2018) and Gupta and Rani (2020) used process data

obtained from Cuckoo that included all processes that a binary created, injected, or terminated. The model by Almutairi et al. (2020) attempted to detect botnets by checking the active time of processes. Pektaş and Acarman (2018) used API calls related to process activities like creating and terminating processes.

Combining a variety of features in a single detector allows hybrid detection of host and network activity. As discussed in section 2.2, this approach can help catch new malware variants: if a malware variant has altered its system-level behavior in an attempt to evade detection, it may still exhibit network behavior that is recognized as malicious, or vice versa. Almutairi et al. (2020) proposed a hybrid model combining network and host data analyzers to detect botnet-specific behavior and reached a high accuracy of 99.6 % in their DT model. Its advantage is that if one of the analyzers fails, the other can still detect it based on other features. Though their model is specific to botnets, the principle of two detectors specialized in different types of behavior can be applied more widely. In another paper, Pektaş and Acarman (2018) monitored different network features like DNS requests and destination ports through the API calls related to them. They combined both host and network features to the same online-learning-based model but still achieved a high accuracy of 98 %.

Whether the features are incorporated into one model, or used to form an ensemble detector with multiple submodules, considering several features is generally beneficial for the detector's accuracy. Especially a hybrid approach targeting both system and network behavior is based on two very different types of behavior which could be advantageous in catching evasive malware. However, including more features in a model increases the complexity which should be taken into account in feature selection.

4.6.5 Solutions for common detection challenges

Due to the nature of the research questions and the literature search approach, all the reviewed papers applied behavior-based techniques at least partially in their

proposed models. The papers that used static features applied them as general rules instead of creating signatures from them, with the exception of the denylists used by Ghafir et al. (2018). For research interests the behavior-based approach naturally offers a lot more possibilities to explore when building an effective detector. Therefore the challenges addressed in this section are all related to behavior-based detection.

As discussed earlier in section 2.2.2, a major downside of behavior-based techniques is that they require much more computational resources compared to signature-based detectors, particularly in machine learning applications. ML and especially DL models require the heaviest computation during the training phase. However, updating and retraining the model with new samples is less demanding, so heavy initial training requirements do not necessarily mean that a model is impractical to use. The training and detection times are dependent on various factors like the selected technique and the format of the sample data. For example, kNN models do not need to be trained to the same extent as other ML methods but their detection speed is much slower (Zhang et al. 2019). The size of the training dataset also has a significant impact on training time and can heavily influence the detection accuracy. The model by Zhang et al. (2019) took about a day and a half to train with 4,000 samples, while in the online learning algorithm model by Pektaş and Acarman (2018) the initial analysis of 60,000 samples took 15 days. However, it was not always clear what kind of training setup was used e.g. in terms of the CPU and memory resources of the computers.

Some of the fastest detection times were reported by Zhang et al. (2019) who achieved a detection speed of 0.034 s with their CNN-BPNN detector and still maintained a detection accuracy of over 95 %. To attain this, they used principal component analysis which is a method to reduce the dimensionality of large datasets to smaller ones that retain key information from the large set. Kishore, Barisal, and Mohapatra (2020a) also reported a detection speed of 0.03 s in their model combining SVM and HTM detectors with a similar accuracy as Zhang et al. (2019). The effect of data type can be seen in the study by Kishore, Barisal, and Mohapatra

(2020b), whose model—while reaching a higher accuracy of 99.4 %—had a slightly slower detection time of 1.03 s caused by their approach of representing samples in image format. Nevertheless, the results are promising and show that the trade-off between accuracy and detection speed is not an either-or situation, and a functional balance between them can be found.

Many of the analyzed papers did not discuss the performance requirements of their proposed model, or how applicable it is for practical antivirus solutions. Alageel and Maffeis (2021) simply stated that solving the performance issues is excluded from their scope but several papers did not address the issue at all. The solutions proposed in the papers may be largely theoretical due to the academic approach and solving practical performance issues is frequently left for future research.

The availability of balanced datasets has been identified as an issue for ML models in the past. Most of the reviewed papers chose to use balanced datasets with roughly equal numbers of benign and malicious samples. Imbalanced datasets were also used in the favor of both malicious and benign samples. A higher rate of benign samples reflects real-life situations more accurately but based on the reviewed papers there was no clear correlation between the malicious-to-benign sample ratio and accuracy. The studies also used datasets of varying sizes—Íncer Romeo et al. (2018) trained their model with 500,000 and validated with 40,000 samples, while Wijaya, Lim, and Kotualubun (2022) used 480 malware samples for training and 160 for testing. Anomaly-based solutions like the one by Tajoddin and Abadi (2019) used only benign data for training. Most papers used publicly available datasets but Alageel and Maffeis (2021) collected their own data on APT C&C domains themselves and released the dataset with their research. Based on the variety of datasets used it seems that balanced training data is readily available.

From the perspective of concept drift, the issue of datasets seems more pressing. Li et al. (2022) pointed out the effect of concept drift by testing their model (trained with a dataset form 2019) with two datasets from 2020, one from the first half and one from the second half of the year. The model's original accuracy 98.43 % decreased by almost 6 %, and the false negative rate increased from the initial 0.84 % to 9.18 %. It is

clear that even in a short amount of time the model is impacted significantly by new malware variants introducing features that were not learned by the original model. Chen et al. (2020) suggest that an anomaly-based approach may be advantageous against this issue in the case of botnet detection. However, concept drift affects both benign and malicious binaries so anomaly-based detection alone is not a sufficient solution (Raff and Nicholas 2020). As a basic principle, generalizing malicious behavior is advantageous to detecting new variants but it may cause losing some of the accuracy. Periodically updating the models with fresh samples is likely the most straightforward way to keep them up-to-date.

A key limitation of behavior-based solutions is that some malware can detect it is executing in a sandbox environment and terminate itself to evade detection. This was identified also in the experiment by Pektaş and Acarman (2018) who reported that during their analysis 25 % of the malware samples did not run. Additionally, 45 % of the samples did not perform enough activities to provide useful data because they required user interaction to be installed, or more CPU and memory to run. Samples that fail to run decrease the accuracy of the detector which may lead to malware being allowed to run in the host system after passing detection. Continuing to monitor activity in the host could help catching malware that has already installed, like how Almutairi et al. (2020) monitored the active time of processes and modifications on critical API keys to detect botnets in an early phase. As an alternate method, static extraction of API calls can be used to complement dynamic techniques, though the approach is defenseless against encryption and other obfuscation methods that prevent malware from being disassembled.

Behavior-based techniques are by principle resistant to code obfuscation but there are still ways for malware authors to deceive detectors by altering the malware's behavior. Detection based on API calls is a common technique but detectors can be confused with redundant API calls that mask behavioral patterns in the malware. The approach proposed by D'Angelo, Ficco, and Palmieri (2021) counteracts this by using the transition probabilities between API invocations in the call sequence. He and Kim (2019), on the other hand, generated adversary samples with redundant

API injection and tested their model against them. The grayscale images proved to be resilient against API injection but the performance of RGB images suffered more noticeably as the number of injected calls increased.

There are still various challenges related to malware detection and behavior-based solutions. The most significant limitations are related to computational resources, sandbox evasion, and concept drift. Detection models need to be designed with the hardware limitations in mind. The computational performance is impacted by the selection and processing of the features, and the selection and application of the detection technique. Combination of static and behavioral features is likely necessary to mitigate both sandbox evasion and code obfuscation. In machine learning applications the models need to be updated regularly to minimize the effect of concept drift and to maintain a high detection accuracy.

5 Conclusions

This thesis presents a systematic review of detection techniques for common malware features. The topic is of active interest in the academic field which was reflected by the vast number of relevant search results. The study was narrowed down to a total of 33 papers published between 2018 and 2023 which were analyzed to identify common techniques related to feature extraction, feature processing, and malware detection.

As a prerequisite for conducting the systematic review, common malware features were first reviewed. Eight malware families were selected from the most prevalent malware of 2022, and techniques used by them were gathered from various electronic sources like blog posts and white papers published by commercial security providers. The most common techniques were identified as modifying registry entries, scheduling tasks, process injection, creating processes, and contacting a C&C server. Additionally, API calls were recognized as a key feature since they are used to apply many of the aforementioned techniques.

The features were used in the literature search to narrow down the search results. During the review process, many of the techniques identified earlier turned out to not be relevant for malware detection, or they were detected through API call activity. Accordingly, API calls were the most studied feature among the sample papers. They are a versatile feature as they can reflect various different malware behaviors. C&C communication was another well-represented feature and the papers studied several different behaviors and techniques related to it. Targeting both API calls and C&C communication enables covering a range of malware activity and is advantageous for ideal results, as combining different features generally benefits the detection accuracy.

The majority of the papers applied machine learning in their proposal across all the studied features. Models based on both classical ML and DL techniques performed well and were suitable for datasets combining a variety of features. The non-

ML techniques featured in the papers were denylists applied to detecting C&C communication, and statistical and stochastic models used in API call detection. Notably all the papers applied behavior-based detection which is preferable to the signature-based approach and even necessary in order to detect new malware variants.

Many studies built ensemble models with several submodules that specialized in detecting specific features. An ensemble model can be set up to verify samples incrementally by sending samples initially classified as benign to following detectors to reanalyze them. Alternatively, the ensemble can consist of independent detectors that all monitor the samples concurrently. The submodules form a safety net so that if malware passes through one detector, another one may still catch it. The ensemble approach is also a logical way to build hybrid models where system and network detection are separated to different modules. As a drawback, it requires more computational resources due to the added complexity.

The literature review highlighted possible solutions to common detection challenges like detection speed, evasive tactics, and the availability of datasets. The review showed that achieving fast detection times while retaining a high accuracy is attainable with suitable feature processing combined with an ML model. As an example of evasion mitigation, different techniques were used to counteract redundant API call injection. The papers featured a variety of large datasets of both malware and benign programs which indicates that there are plenty of resources to train ML models for malware detection. However, concept drift makes datasets quickly out of date which increases the need to update models regularly.

Many of the challenges related to malware detection were still left with no definitive solutions. A large number of the papers did not discuss the computational requirements of their proposed model which is a key factor when building practical antivirus solutions, especially in the case of behavior-based ML models. Though some evasive techniques were addressed in the papers, others like sandbox evasion were not thoroughly resolved. Sandbox evasion is a major drawback related to the behavior-based approach and can significantly reduce the accuracy of detectors.

The goal of this thesis was to provide a cross section of the current malware features and detection techniques. It was conducted with a limited scope due to the nature of a Master's thesis completed by a single author. Having only one person conduct the selection and data extraction processes may skew the results, and the number of selected papers and the depth of the analysis were restricted by time constraints. Despite its limitations, the study successfully identified features of some of the most widespread malware families today, presented novel studies that target detecting those features, and highlighted techniques that can be applied for accurate and practical detection solutions.

This thesis examined features of the most prevalent malware families of recent years. Many of the inspected malware families originated years ago and thus do not necessarily represent the newest challenges in malware detection. An interesting topic for further research would be to study novel malware techniques and ways to counteract them.

Bibliography

Aboaoja, Faitouri A, Anazida Zainal, Fuad A Ghaleb, Bander Ali Saleh Al-rimy, Taiseer Abdalla Elfadil Eisa, and Asma Abbas Hassan Elnour. 2022. "Malware Detection Issues, Challenges, and Future Directions: A Survey". *Applied Sciences* 12 (17): 8482.

Alageel, Almuthanna, and Sergio Maffeis. 2021. "Hawk-Eye: holistic detection of APT command and control domains". In *Proceedings of the 36th annual ACM symposium on applied computing*, 1664–1673.

Almutairi, Suzan, Saoucene Mahfoudh, Sultan Almutairi, and Jalal S Alowibdi. 2020. "Hybrid Botnet Detection Based on Host and Network Analysis". *Journal of Computer Networks and Communications* 2020:1–16.

Amer, Eslam, Ivan Zelinka, and Shaker El-Sappagh. 2021. "A Multi-Perspective malware detection approach through behavioral fusion of API call sequence". *Computers & Security* 110:102449.

Ang, Miguel Carlo. May 2019. *Trickbot Watch: Arrival via Redirection URL in Spam.* Trend Micro. Visited on February 8, 2023. `https://www.trendmicro.com/en_us/research/19/e/trickbot-watch-arrival-via-redirection-url-in-spam.html`.

Biolchini, Jorge, Paula Gomes Mian, Ana Candida Cruz Natali, and Guilherme Horta Travassos. 2005. *Systematic Review in Software Engineering.* Technical report. System Engineering and Computer Science Department, COPPE/UFRJ.

Brereton, Pearl, Barbara A Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. "Lessons from applying the systematic literature review process within the software engineering domain". *Journal of Systems and Software* 80 (4): 571–583.

Campbell, Ryan, and Devin Cargill. July 2020. *Automating Remote Remediation of TrickBot via Falcon's Real Time Response API: Part 1.* CrowdStrike. Visited on February 23, 2023. https://www.crowdstrike.com/blog/automating-remote-remediation-of-trickbot-part-1/.

Check Point. January 2021. *Attacks targeting healthcare organizations spike globally as COVID-19 cases rise again.* Visited on September 22, 2022. https://blog.checkpoint.com/2021/01/05/attacks-targeting-healthcare-organizations-spike-globally-as-covid-19-cases-rise-again/.

———. February 2022. *January 2022's Most Wanted Malware: Lokibot Returns to the Index and Emotet Regains Top Spot.* Visited on February 8, 2023. https://blog.checkpoint.com/2022/02/08/january-2022s-most-wanted-malware-lokibot-returns-to-the-index-and-emotet-regains-top-spot/.

———. January 2023. *December 2022's Most Wanted Malware: Glupteba Entering Top Ten and Qbot in First Place.* Visited on February 8, 2023. https://blog.checkpoint.com/2023/01/13/december-2022s-most-wanted-malware-glupteba-entering-top-ten-and-qbot-in-first-place/.

———. n.d. *XMRig Malware.* Visited on February 10, 2023. https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-malware/xmrig-malware/.

Chen, Chao. November 2012. *Ramnit bot.* Virus Bulletin. Visited on February 15, 2023. https://www.virusbulletin.com/virusbulletin/2012/11/ramnit-bot.

Chen, Tao, Guangming Zhou, Zhangpu Liu, and Tao Jing. 2020. "A Novel Ensemble Anomaly based Approach for Command and Control Channel Detection". In *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, 74–78.

Chen, Zhenshuo, Eoin Brophy, and Tomas Ward. 2021. "Malware Classification Using Static Disassembly and Machine Learning". In *The 29th Irish Conference on Artificial Intelligence and Cognitive Science 2021,* 48–59. School of Computer Science, University College Dublin.

CISA. January 2020. *Alert (TA18-201A): Emotet Malware.* Visited on February 16, 2023. https://www.cisa.gov/uscert/ncas/alerts/TA18-201A.

Cybereason. September 2021. *Threat Analysis Report: PrintNightmare and Magniber Ransomware.* Visited on February 17, 2023. https://www.cybereason.com/blog/threat-analysis-report-printnightmare-and-magniber-ransomware.

Cybersecurity and Infrastructure Security Agency, Australian Cyber Security Centre. August 2022. "2021 Top Malware Strains". https://www.cisa.gov/uscert/sites/default/files/publications/aa22-216a-2021-top-malware-strains.pdf.

D'Angelo, Gianni, Massimo Ficco, and Francesco Palmieri. 2021. "Association rule-based malware classification using common subsequences of API calls". *Applied Soft Computing* 105:107234.

David, Omid E, and Nathan S Netanyahu. 2015. "Deepsign: Deep learning for automatic malware signature generation and classification". In *2015 International Joint Conference on Neural Networks (IJCNN),* 1–8. IEEE.

Deng, Xiyue, and Jelena Mirkovic. 2018. "Malware Analysis Through High-level Behavior". In *11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18).*

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *arXiv preprint arXiv:1810.04805.*

Digital Intelligence. 2021. *The COVID Crime Index 2021.* BAE Systems. https://www.baesystems.com/en-financialservices/insights/the-covid-crime-index.

Ding, Zhenquan, Hui Xu, Yonghe Guo, Longchuan Yan, Lei Cui, and Zhiyu Hao. 2022. "Mal-Bert-GCN: Malware Detection by Combining Bert and GCN". In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications* (*TrustCom*), 175–183. IEEE.

Dunn, John E. August 2011. *Warning After Zeus Bank Trojan Fused with Ramnit Worm.* CSO. Visited on February 6, 2023. https://www.csoonline.com/article/2129360/warning-after-zeus-bank-trojan-fused-with-ramnit-worm.html.

El Merabet, Hoda, and Abderrahmane Hajraoui. 2019. "A Survey of Malware Detection Techniques Based on Machine Learning". *International Journal of Advanced Computer Science and Applications* 10 (1).

F-Secure. n.d.(a). *Cryptolocker.* Visited on October 6, 2022. https://www.f-secure.com/v-descs/trojan_w32_cryptolocker.shtml.

———. n.d.(b). *Trojan-Spy:W32/Zbot.* Visited on February 7, 2023. https://www.f-secure.com/v-descs/trojan-spy_w32_zbot.shtml.

———. n.d.(c). *Trojan.Agent.Formbook.* Visited on February 9, 2023. https://www.f-secure.com/v-descs/trojan_agent_formbook.shtml.

———. n.d.(d). *Useful online security tips and articles.* Visited on January 20, 2023. https://www.f-secure.com/en/home/articles.

FortiGuard Labs. 2022. *Global Threat Landscape Report: A Semiannual Report by FortiGuard Labs, 1H 2022.* Fortinet. https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/threat-report-1h-2022.pdf.

Fox, Neil. May 2021. *Cuckoo Sandbox Overview.* Varonis. Visited on January 13, 2023. https://www.varonis.com/blog/cuckoo-sandbox.

García, David Escudero, Noemí DeCastro-García, and Angel Luis Muñoz Castañeda. 2023. "An effectiveness analysis of transfer learning for the concept drift problem in malware detection". *Expert Systems with Applications* 212:118724.

Ghafir, Ibrahim, Vaclav Prenosil, Mohammad Hammoudeh, Thar Baker, Sohail Jabbar, Shehzad Khalid, and Sardar Jaf. 2018. "BotDet: A system for real time botnet command and control traffic detection". *IEEE Access* 6:38947–38958.

Gibert, Daniel, Carles Mateu, and Jordi Planes. 2020. "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges". *Journal of Network and Computer Applications* 153:102526.

Goyal, Manish, and Raman Kumar. 2020. "The Pipeline Process of Signature-based and Behavior-based Malware Detection". In *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, 497–502. IEEE.

Griffin, Kent, Scott Schneider, Xin Hu, and Tzi-cker Chiueh. 2009. "Automatic generation of string signatures for malware detection". In *International workshop on recent advances in intrusion detection*, 101–120. Springer.

Gupta, Deepak, and Rinkle Rani. 2020. "Improving malware detection using big data and ensemble learning". *Computers & Electrical Engineering* 86:106729.

Hassen, Mehadi, Marco M Carvalho, and Philip K Chan. 2017. "Malware Classification Using Static Analysis Based Features". In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7. IEEE.

He, Ke, and Dong-Seong Kim. 2019. "Malware Detection with Malware Images using Deep Learning Techniques". In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 95–102. IEEE.

Íncer Romeo, Íñigo, Michael Theodorides, Sadia Afroz, and David Wagner. 2018. "Adversarially Robust Malware Detection Using Monotonic Classification". In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, 54–63.

Kaspersky. December 2022. *Kaspersky Security Bulletin 2022.* Visited on February 6, 2023. https://securelist.com/ksb-2022-statistics/108129/.

Kaspersky Lab. 2017. *Fileless attacks against enterprise networks.* Visited on January 20, 2023. https://content.kaspersky-labs.com/se/media//en/business-security/fileless-attacks-against-enterprise-networks.pdf.

———. n.d. *Glossary.* Visited on January 20, 2023. https://encyclopedia.kaspersky.com/glossary/.

Kimayong, Paul. April 2021. *Sysrv Botnet Expands and Gains Persistence.* Juniper. Visited on February 10, 2023. https://blogs.juniper.net/en-us/threat-research/sysrv-botnet-expands-and-gains-persistence.

Kishore, Pushkar, Swadhin Kumar Barisal, and Durga Prasad Mohapatra. 2020a. "An incremental malware detection model for meta-feature API and system call sequence". In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, 629–638. IEEE.

———. 2020b. "Familial analysis of interface and program targeting noise contained malware using image processing". In *2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, 1–6. IEEE.

Kitchenham, Barbara, and Stuart M. Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3.* Technical report. School of Computer Science and Mathematics, Keele University.

Li, Ce, Zijun Cheng, He Zhu, Leiqi Wang, Qiujian Lü, Yan Wang, Ning Li, and Degang Sun. 2022. "DMalNet: Dynamic malware analysis based on API feature engineering and graph learning". *Computers & Security* 122:102872.

Li, Mingxuan, Shichao Lü, and Zhiqiang Shi. 2020. "Malware Detection for Industrial Internet Based on GAN". In *2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, 1:475–481. IEEE.

Lin, Ying-Dar, Yuan-Cheng Lai, Chun-Nan Lu, Peng-Kai Hsu, and Chia-Yin Lee. 2015. "Three-phase behavior-based detection and classification of known and unknown malware". *Security and Communication Networks* 8 (11): 2004–2015.

Llimos, Noel Anthony, and Carl Maverick Pascual. November 2018. *Trickbot Shows Off New Trick: Password Grabber Module.* Trend Micro. Visited on March 1, 2023. https://www.trendmicro.com/en_us/research/18/k/trickbot-shows-off-new-trick-password-grabber-module.html.

Malwarebytes. n.d.(a). *Cybersecurity basics & protection.* Visited on January 20, 2023. https://www.malwarebytes.com/cybersecurity.

———. n.d.(b). *WannaCry.* Visited on October 6, 2022. https://www.malwarebytes.com/wannacry.

Maniriho, Pascal, Abdun Naser Mahmood, and Mohammad Jabed Morshed Chowdhury. 2022a. "A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges". *Future Generation Computer Systems* 130:1–18.

———. 2022b. "MalDetConv: Automated Behaviour-based Malware Detection Framework Based on Natural Language Processing and Deep Learning Techniques". *arXiv preprint arXiv:2209.03547.*

Manna-Browne, Elise. December 2022. *A Threat Hunter's Breakdown Of Magniber Ransomware.* Innovate Cybersecurity. Visited on February 17, 2023. https://innovatecybersecurity.com/news/a-threat-hunters-breakdown-of-magniber-ransomware/.

Markel, Zane, and Michael Bilzor. 2014. "Building a machine learning classifier for malware detection". In *2014 Second Workshop on Anti-malware Testing Research (WATeR)*, 1–4. IEEE.

Masubuchi, Yuma. May 2021. *Alert (TA18-201A): Emotet Malware.* JPCERT/CC. Visited on February 16, 2023. https://blogs.jpcert.or.jp/en/2021/05/xmrig.html#2.

Microsoft Learn. February 2022. *Run and RunOnce Registry Keys.* Microsoft. Visited on February 23, 2023. https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys.

MITRE ATT&CK. June 2022a. *Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder.* Visited on February 23, 2023. https://attack.mitre.org/techniques/T1547/001/.

———. June 2022b. *Create or Modify System Process: Windows Service.* Visited on February 24, 2023. https://attack.mitre.org/techniques/T1543/003/.

———. April 2022c. *Native API.* Visited on April 20, 2023. https://attack.mitre.org/techniques/T1106/.

———. October 2022d. *Process Injection.* Visited on March 1, 2023. https://attack.mitre.org/techniques/T1055/.

———. July 2022e. *Scheduled Task/Job: Scheduled Task.* Visited on February 23, 2023. https://attack.mitre.org/techniques/T1053/005/.

Mohaisen, Aziz, Omar Alrawi, and Manar Mohaisen. 2015. "AMAL: high-fidelity, behavior-based automated malware analysis and classification". *Computers & Security* 52:251–266.

Moser, Andreas, Christopher Kruegel, and Engin Kirda. 2007. "Limits of static analysis for malware detection". In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 421–430. IEEE.

Mosli, Rayan, Rui Li, Bo Yuan, and Yin Pan. 2017. "A behavior-based approach for malware detection". In *IFIP International Conference on Digital Forensics*, 187–201. Springer.

Moussaileb, Routa, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Bouder. 2021. "A Survey on Windows-based Ransomware Taxonomy and Detection Mechanisms: Case Closed?" *ACM Computing Surveys (CSUR)* 54 (6): 1–36.

Najafabadi, Maryam M, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. 2015. "Deep learning applications and challenges in big data analytics". *Journal of Big Data* 2 (1): 1–21.

Ng, Chee Keong, Frank Jiang, Leo Yu Zhang, and Wanlei Zhou. 2019. "Static malware clustering using enhanced deep embedding method". *Concurrency and Computation: Practice and Experience* 31 (19): e5234.

Novo, Carlos, and Ricardo Morla. 2020. "Flow-based Detection and Proxy-based Evasion of Encrypted Malware C2 Traffic". In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, 83–91.

O'Kane, Philip, Sakir Sezer, and Kieran McLaughlin. 2011. "Obfuscation: The hidden malware". *IEEE Security & Privacy* 9 (5): 41–47.

Pektaş, Abdurrahman, and Tankut Acarman. 2018. "Malware classification based on API calls and behaviour analysis". *IET Information Security* 12 (2): 107–117.

Perdisci, Roberto, Wenke Lee, and Nick Feamster. 2010. "Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces." In *NSDI*, 10:14.

Petersen, Kai, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. "Systematic Mapping Studies in Software Engineering". In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, 1–10.

Piet, Julien, Blake Anderson, and David McGrew. 2018. "An In-Depth Study of Open-Source Command and Control Frameworks". In *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, 1–8. IEEE.

Praszmo, Michał. September 2017. *Ramnit—in-depth analysis.* CERT Polska. Visited on February 15, 2023. `https://cert.pl/en/posts/2017/09/ramnit-in-depth-analysis/`.

Proofpoint. November 2022. *A Comprehensive Look at Emotet Virus' Fall 2022 Return.* Visited on February 9, 2023. `https://www.proofpoint.com/us/blog/threat-insight/comprehensive-look-emotets-fall-2022-return`.

Publication Forum. September 2016. *Publication Forum ceases to classify conferences under the established name of the event.* Visited on April 20, 2023. https://julkaisufoorumi.fi/en/news/publication-forum-ceases-classify-conferences-under-established-name-event.

———. March 2023. *Classification criteria.* Visited on April 19, 2023. https://julkaisufoorumi.fi/en/evaluations/classification-criteria.

Rabadi, Dima, and Sin G Teo. 2020. "Advanced Windows Methods on Malware Detection and Classification". In *Annual Computer Security Applications Conference*, 54–68.

Raff, Edward, and Charles Nicholas. 2020. "A Survey of Machine Learning Methods and Challenges for Windows Malware Classification". In *NeurIPS 2020 Workshop: ML Retrospectives, Surveys & Meta-Analyses.* Neural Information Processing Systems Foundation.

Ravi, Vinayakumar, Mamoun Alazab, Shymalagowri Selvaganapathy, and Rajasekhar Chaganti. 2022. "A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems". *Computer Communications* 195:73–81.

Remillano II, Augusto, Jay Nebre, and Arianne Dela Cruz. December 2019. *Monero Miner Obfuscated via Process Hollowing.* Trend Micro. Visited on February 10, 2023. https://www.trendmicro.com/en_fi/research/19/l/almost-hollow-and-innocent-monero-miner-remains-undetected-via-process-hollowing.html.

Rossow, Christian, Christian J Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten Van Steen. 2012. "Prudent practices for designing malware experiments: Status quo and outlook". In *2012 IEEE Symposium on Security and Privacy*, 65–79. IEEE.

Saad, Sherif, William Briguglio, and Haytham Elmiligi. 2019. "The Curious Case of Machine Learning in Malware Detection". In *Proceedings of the 5th International Conference on Information Systems Security and Privacy*, 528–535. SciTePress.

Salinas, Marc, and José Miguel Holguín. June 2017. *Evolution of Trickbot.* S2 Grupo. https://www.securityartwork.es/wp-content/uploads/2017/07/Trickbot-report-S2-Grupo.pdf.

Schläpfer, Patrick. October 2022. *Magniber Ransomware Adopts JavaScript, Targeting Home Users with Fake Software Updates.* HP. Visited on February 17, 2023. https://threatresearch.ext.hp.com/magniber-ransomware-switches-to-javascript-targeting-home-users-with-fake-software-updates/.

Secarma. n.d. *Process Injection Part 1: The Theory.* Visited on March 1, 2023. https://secarma.com/process-injection-part-1-the-theory/.

Shalaginov, Andrii, Sergii Banin, Ali Dehghantanha, and Katrin Franke. 2018. "Machine Learning Aided Static Malware Analysis: A Survey and Tutorial". *Cyber Threat Intelligence*, 7–45.

Silverio, Adolph, Jeric Abordo, Khristian Morales, and Maria Viray. May 2022. *Bruised but Not Broken: The Resurgence of the Emotet Botnet Malware.* Trend Micro. Visited on February 9, 2023. https://www.trendmicro.com/en_us/research/22/e/bruised-but-not-broken--the-resurgence-of-the-emotet-botnet-malw.html.

Singh, Jagsir, and Jaswinder Singh. 2021. "A survey on machine learning-based malware detection in executable files". *Journal of Systems Architecture* 112:101861.

Souri, Alireza, and Rahil Hosseini. 2018. "A state-of-the-art survey of malware detection approaches using data mining techniques". *Human-centric Computing and Information Sciences* 8 (1): 1–22.

Splunk Threat Research Team. November 2022. *Inside the Mind of a 'Rat'—Agent Tesla Detection and Analysis.* Splunk. Visited on February 17, 2023. https://www.splunk.com/en_us/blog/security/inside-the-mind-of-a-rat-agent-tesla-detection-and-analysis.html.

Spooren, Jan, Davy Preuveneers, Lieven Desmet, Peter Janssen, and Wouter Joosen. 2019. "On the use of DGAs in malware: an everlasting competition of detection and evasion". *ACM SIGAPP Applied Computing Review* 19 (2): 31–43.

Staples, Mark, and Mahmood Niazi. 2007. "Experiences using systematic review guidelines". Evaluation and Assessment in Software Engineering, *Journal of Systems and Software* 80 (9): 1425–1437.

Symantec. July 2018. *The Evolution of Emotet: From Banking Trojan to Threat Distributor.* Visited on February 9, 2023. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/evolution-emotet-trojan-distributor.

Szőr, Péter. 2005. *The Art of Computer Virus Research and Defense.* Addison-Wesley Professional. ISBN: 0321304543.

Tajoddin, Asghar, and Mahdi Abadi. 2019. "RAMD: registry-based anomaly malware detection using one-class ensemble classifiers". *Applied Intelligence* 49:2641–2658.

Tayyab, Umm-e-Hani, Faiza Babar Khan, Muhammad Hanif Durad, Asifullah Khan, and Yeon Soo Lee. 2022. "A Survey of the Recent Trends in Deep Learning Based Malware Detection". *Journal of Cybersecurity and Privacy* 2 (4): 800–829.

Threatpost. February 2021. *Agent Tesla Trojan 'Kneecaps' Microsoft's Anti-Malware Interface.* Visited on September 16, 2022. https://threatpost.com/agent-tesla-microsoft-asmi/163581/.

Trend Micro. September 2014. *RAMNIT.* Visited on February 6, 2023. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/RAMNIT/.

———. June 2015. *ZEUS.* Visited on February 7, 2023. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ZEUS/.

Trend Micro. May 2017. *After WannaCry, UIWIX Ransomware Follows Suit.* Visited on January 26, 2023. https://www.trendmicro.com/en_fi/research/17/e/wannacry-uiwix-ransomware-monero-mining-malware-follow-suit.html.

Trend Micro Research. January 2023. *Ransomware Spotlight: Magniber.* Trend Micro. Visited on February 7, 2023. https://www.trendmicro.com/vinfo/fi/security/news/ransomware-spotlight/ransomware-spotlight-magniber.

———. n.d. *Exploring Emotet's Activities.* Visited on February 9, 2023. https://documents.trendmicro.com/assets/white_papers/ExploringEmotetsActivities_Final.pdf.

Vinayakumar, R, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, and Sitalakshmi Venkatraman. 2019. "Robust intelligent malware detection using deep learning". *IEEE Access* 7:46717–46738.

Walter, Jim. April 2021. *Agent Tesla | Old RAT Uses New Tricks to Stay on Top.* SentinelLabs. Visited on February 9, 2023. https://www.sentinelone.com/labs/agent-tesla-old-rat-uses-new-tricks-to-stay-on-top/.

Wijaya, Andre, Charles Lim, and Yohanes Syailendra Kotualubun. 2022. "Malware Classification Method Using API Call Categorization". In *Proceedings of the 2022 International Conference on Engineering and Information Technology for Sustainable Industry*, 1–6.

Wu, Di, Peiqi Guo, and Peng Wang. 2020. "Malware Detection based on Cascading XGBoost and Cost Sensitive". In *2020 International Conference on Computer Communication and Network Security (CCNS)*, 201–205. IEEE.

Xu, Rongheng, Jilin Zhang, and Li Zhou. 2022. "Malware API Sequence Detection Model based on Pre-trained BERT in Professional domain". In *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 1059–1060. IEEE.

You, Ilsun, and Kangbin Yim. 2010. "Malware obfuscation techniques: A brief survey". In *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, 297–300. IEEE.

Zhang, Jixin, Zheng Qin, Hui Yin, Lu Ou, and Kehuan Zhang. 2019. "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding". *Computers & Security* 84:376–392.

Zhang, Xiaopeng. June 2017. *In-Depth Analysis of A New Variant of .NET Malware AgentTesla.* Fortinet. Visited on February 9, 2023. https://www.fortinet.com/blog/threat-research/in-depth-analysis-of-net-malware-javaupdtr.

———. April 2021a. *Deep Analysis: FormBook New Variant Delivered in Phishing Campaign – Part II.* Fortinet. Visited on February 9, 2023. https://www.fortinet.com/blog/threat-research/deep-analysis-formbook-new-variant-delivered-phishing-campaign-part-ii.

———. December 2021b. *Phishing Campaign Targeting Korean to Deliver Agent Tesla New Variant.* Fortinet. Visited on February 23, 2023. https://www.fortinet.com/blog/threat-research/phishing-campaign-targeting-korean-to-deliver-agent-tesla-new-variant.

———. January 2023. *Phishing Campaign Targeting Korean to Deliver Agent Tesla New Variant.* Fortinet. Visited on April 20, 2023. https://www.fortinet.com/blog/threat-research/malicious-code-cryptojacks-device-to-mine-for-monero-crypto.

Appendices

A Accepted papers

Table A1: Analyzed papers

Authors	Title	Year
Ghafir et al.	BotDet: A System for Real Time Botnet Command and Control Traffic Detection	2018
Íncer Romeo et al.	Adversarially Robust Malware Detection Using Monotonic Classification	2018
Pektaş and Acarman	Malware classification based on API calls and behaviour analysis	2018
Piet, Anderson and McGrew	An In-Depth Study of Open-Source Command and Control Frameworks	2018
Yin et al.	A malware detection system based on heterogeneous information network	2018
He and Kim	Malware Detection with Malware Images using Deep Learning Techniques	2019
Liu and Wang	A Robust Malware Detection System Using Deep Learning on API Calls	2019
Lu et al.	ASSCA: API sequence and statistics features combined architecture for malware detection	2019
Spooren et al.	On the use of DGAs in malware: an everlasting competition of detection and evasion	2019
Tajoddin and Abadi	RAMD: registry-based anomaly malware detection using one-class ensemble classifiers	2019
Zhang et al.	A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding	2019
Almutairi et al.	Hybrid Botnet Detection Based on Host and Network Analysis	2020
Chen et al.	A Novel Ensemble Anomaly based Approach for Command and Control Channel Detection	2020
Goyal and Kumar	The Pipeline Process of Signature-based and Behavior-based Malware Detection	2020
Gupta and Rani	Improving malware detection using big data and ensemble learning	2020
Kishore, Barisal and Mohapatra	Familial analysis of interface and program targeting noise contained malware using image processing	2020

Table A1: Analyzed papers (continued)

Authors	**Title**	**Year**
Kishore, Barisal and Mohapatra	An incremental malware detection model for meta-feature API and system call sequence	2020
Li, Lü and Shi	Malware Detection for Industrial Internet Based on GAN	2020
Namani and Khan	Symbolic execution based feature extraction for detection of malware	2020
Novo and Morla	Flow-based Detection and Proxy-based Evasion of Encrypted Malware C2 Traffic	2020
Rabadi and Teo	Advanced Windows Methods on Malware Detection and Classification	2020
Wu, Guo and Wang	Malware Detection based on Cascading XGBoost and Cost Sensitive	2020
Alageel and Maffeis	Hawk-Eye: holistic detection of APT command and control domains	2021
Amer, Zelinka and El-Sappagh	A Multi-Perspective malware detection approach through behavioral fusion of API call sequence	2021
D'Angelo, Ficco and Palmieri	Association rule-based malware classification using common subsequences of API calls	2021
Ding et al.	Mal-Bert-GCN: Malware Detection by Combining Bert and GCN	2022
Li et al.	DMalNet: Dynamic malware analysis based on API feature engineering and graph learning	2022
Namita, Prachi and Sharma	Windows Malware Detection using Machine Learning and TF-IDF Enriched API Calls Information	2022
Ravi et al.	A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems	2022
Wijaya, Lim and Kotualubun	Malware Classification Method Using API Call Categorization	2022
Xu, Zhang and Zhou	Malware API Sequence Detection Model based on Pre-trained BERT in Professional domain	2022
Xue, Wang and Feng	Malicious Network Software Detection Based on API Call	2022
García, DeCastro-García and Castañeda	An effectiveness analysis of transfer learning for the concept drift problem in malware detection	2023

B Discarded papers

Table A2: Discarded papers

Authors	Paper	Year	Reason for discarding
Oyama	Trends of anti-analysis operations of malwares observed in API call logs	2017	outside of time range
Saleh, Li and Xu	Multi-context features for detecting malicious programs	2017	outside of time range
Abdelsalam et al.	Malware Detection in Cloud Infrastructures Using Convolutional Neural Networks	2018	features unclear or not relevant
Albabtain and Yang	The Process of Reverse Engineering GPU Malware and Provide Protection to GPUS	2018	topic not relevant
Cheng et al.	Towards Paving the Way for Large-Scale Windows Malware Analysis	2018	topic not relevant
Copty et al.	Accurate Malware Detection by Extreme Abstraction	2018	features unclear or not relevant
Cozzi et al.	Understanding Linux Malware	2018	topic not relevant
Hampton, Baig and Zeadally	Ransomware behavioural analysis on windows platforms	2018	discarded during random selection
Hsiao and Kao	The static analysis of WannaCry ransomware	2018	topic not relevant
Joshi et al.	Machine Learning Approach for Malware Detection Using Random Forest Classifier on Process List Data Structure	2018	topic not relevant
Jung and Won	Ransomware detection method based on context-aware entropy analysis	2018	features unclear or not relevant
Kao and Hsiao	The dynamic analysis of WannaCry ransomware	2018	topic not relevant
Kono, Phomkeona and Okamura	An Unknown Malware Detection Using Execution Registry Access	2018	discarded during random selection
Lee, Lee and Soh	Trend of Malware Detection Using Deep Learning	2018	topic not relevant
Lu et al.	ASSCA: API based Sequence and Statistics features Combined malware detection Architecture	2018	discarded during random selection
Lungana-Niculescu, Colesa and Oprisa	False Positive Mitigation in Behavioral Malware Detection Using Deep Learning	2018	JUFO level 0
Mira and Huang	Performance Evaluation of String Based Malware Detection Methods	2018	discarded during random selection
Monnappa	Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware	2018	not a journal article or conference paper
Sang, Cuong and Cuong	An Effective Ensemble Deep Learning Framework for Malware Detection	2018	features unclear or not relevant
Sanjay et al.	An Approach to Detect Fileless Malware and Defend its Evasive mechanisms	2018	topic not relevant
Sethi et al.	A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach	2018	features unclear or not relevant
Sewak, Sahay and Rathore	An investigation of a deep learning based malware detection system	2018	features unclear or not relevant
Sruthi et al.	ACTM: API Call Transition Matrix-based Malware Detection Method	2018	discarded during random selection
Wu et al.	Enhancing Machine Learning Based Malware Detection Model by Reinforcement Learning	2018	features unclear or not relevant
Bae, Kim and Im	Automatic hybrid analysis technique to improve botnet code coverage using fake server	2019	features unclear or not relevant

Table A2: Discarded papers (continued)

Authors	Paper	Year	Reason for discarding
Bai and Shi	Malware Detection Method based on Dynamic Variable Length API Sequence	2019	discarded during random selection
Balogh and Mojžiš	New Direction for Malware Detection Using System Features	2019	JUFO level 0
Block and Dewald	Windows Memory Forensics: Detecting (Un)Intentionally Hidden Injected Code by Examining Page Table Entries	2019	features unclear or not relevant
Borrello et al.	The ROP needle: hiding trigger-based injection vectors via code reuse	2019	topic not relevant
Botacin, de Geus and Grégio	"VANILLA" malware: vanishing antiviruses by interleaving layers and layers of attacks	2019	topic not relevant
Case et al.	HookTracer: A System for Automated and Accessible API Hooks Analysis	2019	discarded during random selection
Garg and Yadav	Malware Detection based on API Calls Frequency	2019	discarded during random selection
He and Kim	Malware Detection with Malware Images using Deep Learning Techniques	2019	duplicate
Khanmohammadi, Khoury and Hamou-Lhadj	On the Use of API Calls for Detecting Repackaged Malware Apps: Challenges and Ideas	2019	topic not relevant
Lee, Chang and Im	DGA-based malware detection using DNS traffic analysis	2019	discarded during random selection
Maiorca, Biggio and Giacinto	Towards Adversarial Malware Detection: Lessons Learned from PDF-based Attacks	2019	secondary study
Manavi and Hamzeh	A new approach for malware detection based on evolutionary algorithm	2019	features unclear or not relevant
Menon	Thwarting C2 Communication of DGA-Based Malware using Process-level DNS Traffic Tracking	2019	topic not relevant
Oak et al.	Malware Detection on Highly Imbalanced Data through Sequence Modeling	2019	topic not relevant
Oosthoek and Doerr	SoK: ATT&CK Techniques and Trends in Windows Malware	2019	topic not relevant
Rosli et al.	Clustering Analysis for Malware Behavior Detection using Registry Data	2019	JUFO level 0
Sai et al.	MACA-I: A Malware Detection Technique using Memory Management API Call Mining	2019	JUFO level 0
Sergeev et al.	Combined side-channels malware detection for NFV infrastructure	2019	features unclear or not relevant
Shamshirband and Chronopoulos	A new malware detection system using a high performance-ELM method	2019	topic not relevant
Shan et al.	Malware Detection Method based on Control Flow Analysis	2019	features unclear or not relevant
Sivakorn et al.	Countering Malicious Processes with Process-DNS Association	2019	discarded during random selection
Sun et al.	An Opcode Sequences Analysis Method For Unknown Malware Detection	2019	features unclear or not relevant
Vinayakumar et al.	Robust Intelligent Malware Detection Using Deep Learning	2019	features unclear or not relevant
Wahab et al.	Power & performance optimized hardware classifiers for efficient on-device malware detection	2019	topic not relevant
Zhang et al.	A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding	2019	duplicate

Table A2: Discarded papers (continued)

Authors	Paper	Year	Reason for discarding
Afreen, Aslam and Ahmed	Analysis of Fileless Malware and its Evasive Behavior	2020	topic not relevant
Amer and Zelinka	A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence	2020	discarded during random selection
Cao et al.	On benign features in malware detection	2020	topic not relevant
Chai et al.	LGMal: A Joint Framework Based on Local and Global Features for Malware Detection	2020	topic not relevant
Chen	A Malware Detection Method Based on Rgb Image	2020	features unclear or not relevant
Chen et al.	Malware classification based on heterogeneous information network representation learning	2020	features unclear or not relevant
Feng, Cui and Hu	Detection and classification of malware based on FastText	2020	discarded during random selection
Fu and Lan	Deep Generative Model for Malware Detection	2020	features unclear or not relevant
Goyal and Kumar	Machine Learning for Malware Detection on Balanced and Imbalanced Datasets	2020	discarded during random selection
Guo et al.	Research on Malware Variant Detection Based on Global Texture Features	2020	features unclear or not relevant
Huang et al.	A Method for Windows Malware Detection Based on Deep Learning	2020	discarded during random selection
Irfan et al.	A Malware Detection Framework Based on Forensic and Unsupervised Machine Learning Methodologies	2020	features unclear or not relevant
Kadiyala et al.	Hardware Performance Counter-Based Fine-Grained Malware Detection	2020	features unclear or not relevant
Lee et al.	Fileless cyberattacks: Analysis and classification	2020	topic not relevant
Mavroeidis and Brule	A nonproprietary language for the command and control of cyber defenses – OpenC2	2020	topic not relevant
Mohanta and Saldanha	Code Injection, Process Hollowing, and API Hooking	2020	not a journal article or conference paper
Mohanta and Saldanha	Malware Analysis and Detection Engineering	2020	not a journal article or conference paper
Naaz and Parveen	A PNN based Malign Attack Detection and Classification Model	2020	discarded during random selection
Namani and Khan	Symbolic execution based feature extraction for detection of malware	2020	duplicate
Poudyal and Dasgupta	AI-Powered Ransomware Detection Framework	2020	discarded during random selection
Ramadhan et al.	Forensic Malware Identification Using Naive Bayes Method	2020	features unclear or not relevant
Ren, Chen and Lu	Space Filling Curve Mapping for Malware Detection and Classification	2020	features unclear or not relevant
Suaboot et al.	Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences	2020	discarded during random selection
Verwer et al.	The Robust Malware Detection Challenge and Greedy Random Accelerated Multi-Bit Search	2020	features unclear or not relevant
Wang et al.	You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis	2020	features unclear or not relevant

Table A2: Discarded papers (continued)

Authors	Paper	Year	Reason for discarding
Wang et al.	A Malware Similarity Analysis Method Based on Network Control Structure Graph	2020	topic not relevant, duplicate
Wilkins, Zincir and Zincir-Heywood	Exploring an artificial arms race for malware detection	2020	topic not relevant
Yang et al.	A Real-Time and Adaptive-Learning Malware Detection Method Based on API-Pair Graph	2020	discarded during random selection
Zhang et al.	Scarecrow: Deactivating Evasive Malware via Its Own Evasive Logic	2020	topic not relevant
Alrawi et al.	Forecasting Malware Capabilities From Cyber Attack Memory Images	2021	topic not relevant
Bae and Lee	Easy Data Augmentation for Improved Malware Detection: A Comparative Study	2021	features unclear or not relevant
Baek et al.	Two-Stage Hybrid Malware Detection Using Deep Learning	2021	topic not relevant
Barr-Smith et al.	Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land	2021	topic not relevant
Borana et al.	An Assistive Tool For Fileless Malware Detection	2021	features unclear or not relevant
Botacin, Grégio and Alves	Near-Memory & In-Memory Detection of Fileless Malware	2021	features unclear or not relevant
Chandrakala et al.	Detection and Classification of Malware	2021	features unclear or not relevant
Charan, Anand and Shukla	DMAPT: Study of data mining and machine learning techniques in advanced persistent threat attribution and detection	2021	not a journal article or conference paper
Cheng et al.	Obfuscation-Resilient Executable Payload Extraction From Packed Malware	2021	topic not relevant
Dai et al.	Using IRP and local alignment method to detect distributed malware	2021	features unclear or not relevant
Darem et al.	An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning	2021	discarded during random selection
García and DeCastro-García	Optimal feature configuration for dynamic malware detection	2021	discarded during random selection
Kan et al.	Investigating Labelless Drift Adaptation for Malware Detection	2021	topic not relevant
Karantzas and Patsakis	An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors	2021	topic not relevant
Kim, Lee and Youn	Malicious Behavior Detection Method Using API Sequence in Binary Execution Path	2021	JUFO level 0
Kundu, Anatharaman and Truong-Huu	An Empirical Evaluation of Automated Machine Learning Techniques for Malware Detection	2021	features unclear or not relevant
Landman and Nissim	Deep-Hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments	2021	topic not relevant
Najafi et al.	NLP-based Entity Behavior Analytics for Malware Detection	2021	features unclear or not relevant

Table A2: Discarded papers (continued)

Authors	Paper	Year	Reason for discarding
Panker and Nissim	Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments	2021	topic not relevant
Peng et al.	Semantics aware adversarial malware examples generation for black-box attacks	2021	topic not relevant
Qi et al.	Unsupervised Domain Adaptation for Static Malware Detection based on Gradient Boosting Trees	2021	features unclear or not relevant
Rathore et la.	Are CNN based Malware Detection Models Robust?: Developing Superior Models using Adversarial Attack and Defense	2021	not a journal article or conference paper
Singh, Chaturvedy and Mishra	Multi-View Learning for Repackaged Malware Detection	2021	topic not relevant
Sun et al.	Effective malware detection scheme based on classified behavior graph in IIoT	2021	topic not relevant
Takey et al.	Real Time early Multi Stage Attack Detection	2021	discarded during random selection
Veeraragavan and Nygård	DeCanSec: A Decentralized Architecture for Secure Statistical Computations on Distributed Health Registry Data	2021	topic not relevant
Voronin and Morozov	Analyzing API Sequences for Malware Monitoring Using Machine Learning	2021	topic not relevant
Yang, He and Cui	A GPU memory leakage code defect detection method based on the API calling feature	2021	topic not relevant
Sharma et al.	Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense	2022	topic not relevant
Al-Andoli et al.	Parallel Deep Learning with a hybrid BP-PSO framework for feature extraction and malware classification	2022	features unclear or not relevant
Alsmadi, Al-Ahmad and Alsmadi	Malware analysis and multi-label category detection issues: Ensemble-based approaches	2022	features unclear or not relevant
Amer et al.	Malware Detection Approach Based on the Swarm-Based Behavioural Analysis over API Calling Sequence	2022	discarded during random selection
Botacin et al.	HEAVEN: A Hardware-Enhanced AntiVirus ENgine to accelerate real-time, signature-based malware detection	2022	features unclear or not relevant
Chen et al.	MalPro: Learning on Process-Aware Behaviors for Malware Detection	2022	discarded during random selection
Demirkıran et al.	An ensemble of pre-trained transformer models for imbalanced multiclass malware classification	2022	discarded during random selection
Ding et al.	Mal-Bert-GCN: Malware Detection by Combining Bert and GCN	2022	duplicate
Divakarla, Reddy and Chandrasekaran	A Novel Approach towards Windows Malware Detection System Using Deep Neural Networks	2022	features unclear or not relevant
Divya, Amritha and Viswanathan	A model to detect domain names generated by DGA malware	2022	discarded during random selection
Falana et al.	Mal-Detect: An intelligent visualization approach for malware detection	2022	features unclear or not relevant
Finder, Sheetrit and Nissim	Time-interval temporal patterns can beat and explain the malware	2022	discarded during random selection
Finder, Sheetrit and Nissim	A time-interval-based active learning framework for enhanced PE malware acquisition and detection	2022	features unclear or not relevant
Fu et al.	Encrypted Malware Traffic Detection via Graph-based Network Analysis	2022	features unclear or not relevant

Table A2: Discarded papers (continued)

Authors	Paper	Year	Reason for discarding
Gao et al.	MaliCage: A packed malware family classification framework based on DNN and GAN	2022	features unclear or not relevant
Gençaydin et al.	Benchmark Static API Call Datasets for Malware Family Classification	2022	topic not relevant
Grace and Sughasiny	Behaviour analysis of inter-app communication using a lightweight monitoring app for malware detection	2022	topic not relevant
He et al.	Deep Neural Network and Transfer Learning for Accurate Hardware-Based Zero-Day Malware Detection	2022	features unclear or not relevant
Jing, Wu and Cui	Ensemble dynamic behavior detection method for adversarial malware	2022	discarded during random selection
Kakisim, Gulmez and Sogukpinar	Sequential opcode embedding-based malware detection method	2022	features unclear or not relevant
Kasarapu et al.	CAD-FSL: Code-Aware Data Generation based Few-Shot Learning for Efficient Malware Detection	2022	features unclear or not relevant
Kumar et al.	Malware Detection Classification using Recurrent Neural Network	2022	features unclear or not relevant
Kumar, Janet and Neelakantan	Identification of malware families using stacking of textural features and machine learning	2022	features unclear or not relevant
Kwan	Markov Image with Transfer Learning for Malware Detection and Classification	2022	features unclear or not relevant
Li et al.	A novel deep framework for dynamic malware detection based on API sequence intrinsic features	2022	discarded during random selection
Mira	A Futuristic Appraisal of Malware Detection by Employing Data Mining	2022	secondary study
Muralidharan et al.	File Packing from the Malware Perspective: Techniques, Analysis Approaches, and Directions for Enhancements	2022	secondary study
Nawaz et al.	MalSPM: Metamorphic malware behavior analysis and classification using sequential pattern mining	2022	discarded during random selection
O'Shaughnessy and Sheridan	Image-based malware classification hybrid framework based on space-filling curves	2022	discarded during random selection
Patil et al.	Roadmap of Digital Forensics Investigation Process with Discovery of Tools	2022	not a journal article or conference paper
Pružinec et al.	KUBO: a framework for automated efficacy testing of anti-virus behavioral detection with procedure-based malware emulation	2022	topic not relevant
Qiang, Yang and Jin	Efficient and Robust Malware Detection Based on Control Flow Traces Using Deep Neural Networks	2022	features unclear or not relevant
Qiao et al.	Adversarial malware sample generation method based on the prototype of deep learning detector	2022	topic not relevant
Quoc et al.	Detecting DGA Botnet based on Malware Behavior Analysis	2022	discarded during random selection
Sharma et al.	Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense	2022	topic not relevant
Singh, Borgohain and Kumar	Performance Enhancement of SVM-based ML Malware Detection Model Using Data Preprocessing	2022	features unclear or not relevant
Tay et al.	Towards Robust Detection of PDF-based Malware	2022	not a journal article or conference paper
Touili	Register Automata for Malware Specification	2022	features unclear or not relevant

Table A2: Discarded papers (continued)

Authors	**Paper**	**Year**	**Reason for discarding**
Vouvoutsis, Casino and Patsakis	On the effectiveness of binary emulation in malware classification	2022	discarded during random selection
Wang	Open Challenges of Malware Detection under Concept Drift	2022	not a journal article or conference paper
Wu, Zhang and Kou	A Model for Malware Detection Method based on API call Sequence Clustering	2022	discarded during random selection
Yan et al.	DitDetector: Bimodal Learning based on Deceptive Image and Text for Macro Malware Detection	2022	features unclear or not relevant
Zhang et al.	Slowing Down the Aging of Learning-Based Malware Detectors With API Knowledge	2022	discarded during random selection
Zhao, Kou and Zhang	Online Learning based Self-updating Incremental Malware Detection Model	2022	not a journal article or conference paper
Alaeiyan et al.	Sober: Explores for invasive behaviour of malware	2023	discarded during random selection
Ceschin et al.	Fast & Furious: On the modelling of malware detection as an evolving data stream	2023	topic not relevant
Deng et al.	MCTVD: A malware classification method based on three-channel visualization and deep learning	2023	features unclear or not relevant
Fascí et al.	Disarming visualization-based approaches in malware detection systems	2023	topic not relevant
Kara	Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges	2023	discarded during random selection
Liu et al.	M3F: A novel multi-session and multi-protocol based malware traffic fingerprinting	2023	features unclear or not relevant
Naeem et al.	Development of a Deep Stacked Ensemble With Process Based Volatile Memory Forensics for Platform Independent Malware Detection and Classification	2023	features unclear or not relevant
Sharma et al.	Multi-dimensional Hybrid Bayesian Belief Network Based Approach for APT Malware Detection in Various Systems	2023	no access
Shaukat, Luo and Varadharajan	A novel deep learning-based approach for malware detection	2023	features unclear or not relevant
Tang et al.	BHMDC: A byte and hex n-gram based malware detection and classification method	2023	features unclear or not relevant
Tsafrir et al.	Efficient feature extraction methodologies for unknown MP4-Malware detection using Machine learning algorithms	2023	features unclear or not relevant
Yang et al.	GooseBt: A programmable malware detection framework based on process, file, registry, and COM monitoring	2023	features unclear or not relevant
Yang et al.	RecMaL: Rectify the malware family label via hybrid analysis	2023	topic not relevant
Zhan et al.	AMGmal: Adaptive mask-guided adversarial attack against malware detection with minimal perturbation	2023	topic not relevant