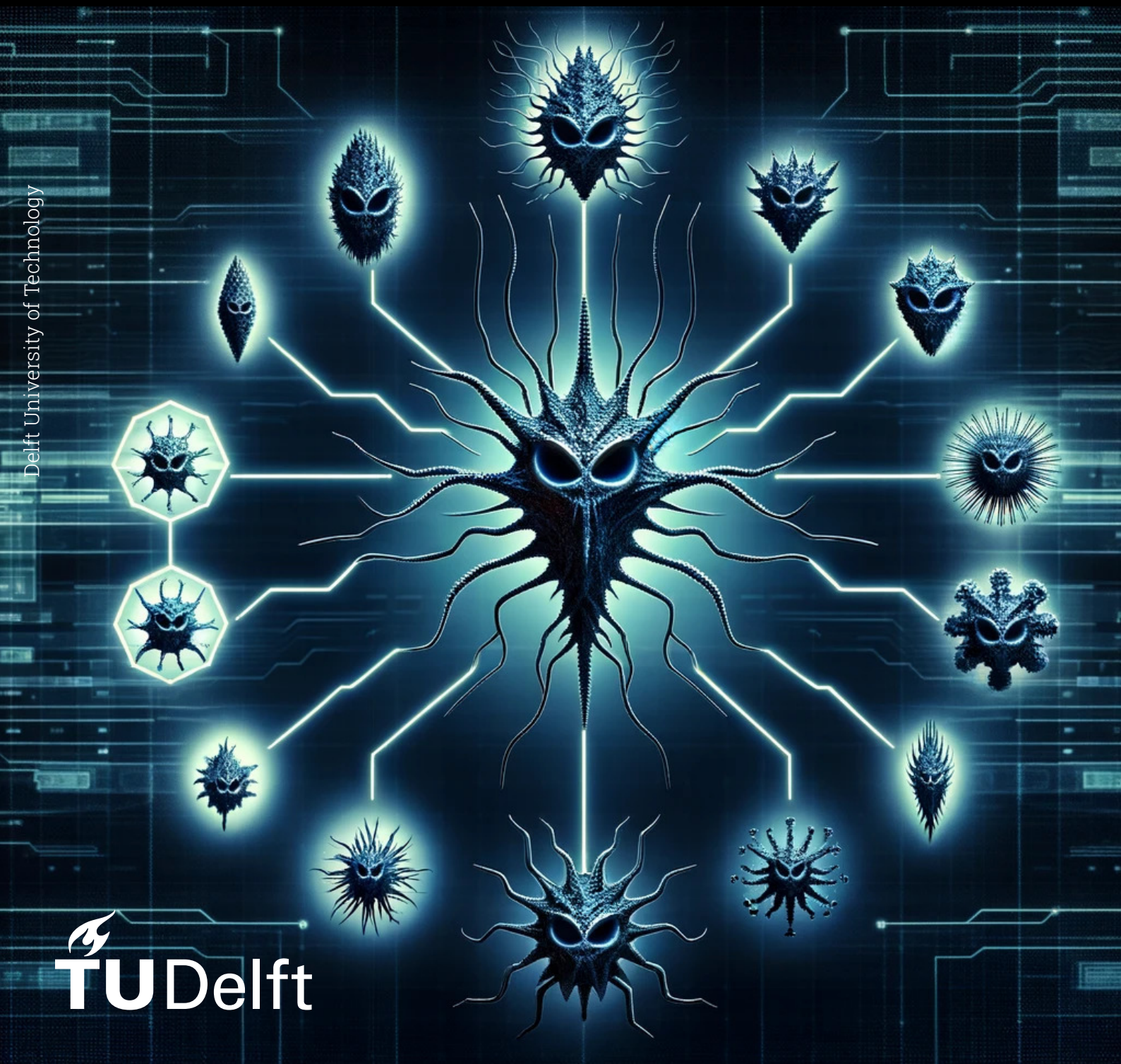
Unveiling the Evolution

Analysing Generational Variances in Malware Families

M.Sc. Thesis

Frank Broy



Delft University of Technology

TU Delft

Unveiling the Evolution

Analysing Generational Variances in Malware Families

by

Frank Broy

to obtain the degree of Master of Science
in Computer Science, with a specialisation in Cybersecurity,
at the Delft University of Technology,
to be defended publicly on Tuesday, June 18, 2024 at 13:00.

Student number:	5016894	
Project duration:	November, 2023 – June, 2024	
Thesis Committee:	Prof. G. Smaragdakis	TU Delft
	Dr. Harm Griffioen	TU Delft
	Dr. Tom Viering	TU Delft
	Stefan Opdebeek	TU Delft

Cover: Image created using the help of DALL-E.

An electronic version of this thesis is available at `https://repository.tudelft.nl/`.

TU Delft

Preface

With this thesis, my academic curriculum comes to an end. During my five years at university, I have met many people from many different backgrounds, and I would lie if I said that not everyone somehow impacted how I evolved and changed as a person. I have made many friends during this time, which I hope will also still stick around for the future.

These five years would not have been possible without the endless support of my parents, who supported me in every decision I made and helped out as much as they could. Even though they probably did not understand everything I did or even studied at university, they still believed in me and were super proud of what I achieved.

I would like to specifically thank Prof. Georgios Smaragdakis for the valuable feedback he gave me while I was working on my thesis. Throughout my curriculum, I have met many professors, and I would like to thank all of them for teaching me what I know today.

A big thank you also goes out to my girlfriend and all of my friends who were there during my time at university, as they helped me through the times when it sometimes got a bit tough. Having someone to turn to for some distraction is essential to keep the focus and motivation on what is important.

*My main takeaway is that you should never stop trying when it gets tough, as everything that follows will feel much easier when you pass the hard challenges. Or, to put it into the words of Patches O'Houlihan: "**If you can dodge a wrench, you can dodge a ball.**"*

Frank Broy
Delft, June 2024

Abstract

The evolution of malware presents an ever-growing challenge to cybersecurity, impacting individuals, organisations, and nations alike. As malicious actors continue to adapt their tactics to bypass security measures, it becomes imperative to understand the evolutionary patterns of malware to stay ahead in the ongoing arms race between defenders and attackers. The complexity and sophistication of modern malware pose significant difficulties in detection and mitigation, making it crucial to unravel its evolving nature to enhance the existing defensive capabilities.

This research focuses on studying the evolutionary dynamics of malware, examining how variants emerge to circumvent existing security measures. Understanding the mechanisms through which malware evolves makes it possible to identify common patterns and develop strategies to predict the behaviour of certain malware. This work mainly encompasses Windows ransomware, particularly the Conti family, with an additional examination of the WannaCry and Ryuk families. The analysis was conducted primarily by applying dynamic malware analysis techniques to the samples. A total of 143 true-positive Conti samples, alongside 75 WannaCry and 21 Ryuk samples, were collected from reputable sources such as VX-Underground and Malware Bazaar. By utilising the ANY.RUN interactive sandbox for dynamic behavioural analysis, malware samples can be executed in a controlled environment and real-time behaviours, such as file modifications or registry changes, can be collected to discern the malware's underlying functionality and potential impact. In addition, the results obtained from Virustotal, a widely-used online malware scanning platform, are considered to get insights into the detection status of the analysed samples across multiple antivirus engines. Finally, Microsoft Defender Antivirus is utilised to classify the variants and eliminate false positives as much as possible. The tactics and techniques outlined in the MITRE ATT&CK Matrix are used to assess sample behaviour. This framework provides valuable insights into the observed behaviour of samples and the methods employed to achieve specific objectives.

The results answer the question "How do different variants of the malware families succeed in bypassing security measures? and split the answer into three smaller ones. Overall, it can be observed that different ransomware share common traits, but differences over time and between variants and families can be seen. Some differences exist between the version of the operating system in which the malware is executed. Malware evolves, and the changes of the malware authors are reflected in their malware's behaviour and structure. Some changes persist, whereas new ways quickly replace others. By understanding the evolution and analysing the patterns that emerge, we can build our defences in a way that predicts incoming threats and creates a safer space for everyone.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Motivation

Malicious software, mostly known as malware, has become an ever-reoccurring threat in the world of technology. Due to the constant evolution of complexity and sophistication of malware, combined with the exponential growth and spread of technology nowadays, the attack landscape for threat actors has increased significantly [37] [4]. One of the main goals of malware is to steal confidential data or to cause damage by encrypting or deleting critical digital assets. However, to achieve this, the malicious software has to bypass existing security measures, such as Antivirus software (AV) or Intrusion Detection Systems (IDS).

The primary motivation behind this research is that AV and IDS detect known malware. Still, even small changes to the functionality and obfuscation methods can enable the malware to evade security measures once more [16]. The defences could be enhanced by gaining insight into how different generations of the same malware family vary in their approaches to bypassing security measures put in place by the industry. Instead of mitigating the harm caused by this malicious software, it could perhaps be prevented altogether.

With each passing day, malicious actors adjust their strategies to circumvent security measures, underscoring the critical need to comprehend the evolutionary patterns of malware. This understanding is crucial in maintaining a competitive edge within the cat-and-mouse game between defenders and attackers. By delving deep into the evolutionary dynamics of malware, this study endeavours to shed light on the processes underlying its evolution and adaptation. Through comprehensive analysis, we aim to discern common patterns and trends, thereby facilitating the development of strategies to enhance the severity and potential impact of specific malware strains.

Despite the extensive research that has already been conducted in this field [9] [8], there remains a gap in comprehending the differences in how various generations and variants of malware bypass existing security measures. The rapid evolution of malware and the evolving threat landscape has led to a lack of consolidated and up-to-date understanding of malware authors' characteristics and evasion strategies [14].

1.2. Research questions

This research aims to bridge the gap by analysing existing knowledge and insights, shedding light on the evolving landscape of malware and their continuous efforts to bypass security measures by answering the main research question: *"How do different generations of the*

malware families succeed in bypassing security measures?". More specifically, the following sub-questions will be looked into in more detail:

1. Are there specific trends in evasion tactics that persist across multiple generations or families?
2. Is there a difference in the behaviour of malware when it is executed on a different version of the operating system? Does this impact the efficiency of the malware?
3. What is the most common behaviour detected across different malware families and their generations, and are different techniques employed to achieve the same goals?

1.3. Contributions

This study brings several essential contributions to the field. First, samples were collected from three notorious malware families: Conti, WannaCry, and Ryuk. These samples were examined using manual and automated methods to pinpoint true positives and eliminate false positives. This dataset is available for further research. Next, these samples were sorted into different variants using Microsoft Defender Antivirus. This classification helps us understand the variations within each malware family and makes it easier to study the differences between variants of the same malware family. Additionally, crafted scripts have been written that automatically search through open-source malware databases, locate samples belonging to a specific malware family, download them, and conduct thorough analyses. Furthermore, a method for comparing different samples from the same malware family has been devised, which allows tracking changes over time and identifying similarities and differences between various variants. Overall, these contributions enhance our understanding of malware behaviour and evolution by giving insights into the constant changes applied by malicious actors and malware authors to beat existing security measures.

1.4. Report Structure

The rest of the thesis is structured to provide a comprehensive and easy-to-follow read about the evolution of malware and its differences over time. After the introduction, Chapter 2 will give some background information to help understand the following sections. In the Background chapter, different malware types are described and specific ways malware can be analysed. Additionally, some related work will be listed in these research areas to complete this thesis. Next, Chapter 3 focuses on the methodology and how this research was conducted so it can be replicated, if necessary. This chapter starts with some safety precautions, followed by the tools used throughout the research. Afterwards, the data collection, pre-processing and analysis process will be described in detail. In Chapter 4, the results will be evaluated and split into multiple categories: general, variant, time, family and environment. In Chapter 5, a discussion of the limitations and problems faced during the research can be found. In Chapter 6, a conclusion will be drawn, where the answers to the research questions will also be repeated, and possible future work will be introduced.

2 Background

Malicious software, mostly known as malware, are computer programs that aim to cause harm in many ways. This can be stealing confidential data, destroying digital assets or simply causing havoc by deleting, destroying or encrypting sensitive information and files. Some of the best-known malware types are described in the following sections, along with their most common features and some examples. Afterwards, there will be an explanation of which malware analysis types exist, and the difference will be made between dynamic and static analysis, as well as their drawbacks and benefits. Following this, executable files on Windows will be analysed and why they are essential in this research, followed by an explanation of the MITRE ATT&CK matrix. At the end of this chapter, some related work and studies will be listed in four different areas.

2.1. Malware

Malware is a term for a collection of different types of malicious software. Many types of malware exist, and all have different behaviours and goals. The following sections describe four of the most well-known malware types, and the final section lists some additional ones.

2.1.1. Ransomware

Ransomware is a cryptography-based malware that encrypts the victim's files and personal data and demands a ransom to decrypt the files and regain access to the machine. The first known ransomware was the "AIDS Trojan" in 1989 [17], which encrypted and munged the names of all directories on the C: drive. It was spread using floppy disks and disguised as having to renew the lease to the PC Cyborg Company. The ransom was only 189 dollars and had to be sent to a PO box in Panama. In the 2010s, ransomware gained significant traction with the rise of cryptocurrencies [36]. Nowadays, ransomware uses cryptocurrency for its ransoms, as these digital currencies are difficult to trace, which makes finding the criminals behind the malware much more difficult. Ransomware became especially famous after attacks such as CryptoLocker in 2013 [26] or WannaCry in 2017.

Ransomware is especially dangerous nowadays, as it is one of the most, if not the most damaging malware. Lately, ransomware attacks have risen both in frequency and scale. In 2023, the damages caused by ransomware have been estimated to be multiple billions of dollars and are predicted to increase even further in the following years [39]. The damage that is caused is much more significant as more and more companies have all their assets digitally, and the individual ransom demands are getting bigger. These ransom demands are some-

times adapted to the companies or individuals they target so the victim can pay them. The financial impact comes from the demanded ransom and the downtime of the infrastructure, reputational damage, and data recovery. The malware authors also target educational facilities, healthcare, and critical infrastructure, which once again underlines this malware's destructive and potentially life-threatening consequences.

Nowadays, ransomware usually spreads through phishing emails, where the victim is tricked into installing a normal-looking attachment or clicking a link. The ransomware uses weaknesses or vulnerabilities in the operating system, such as EternalBlue, which is used by the WannaCry ransomware, and then tries to propagate through the network and infect more machines. With the rise of RaaS, ransomware-as-a-service, the ransomware landscape has been revolutionised again, making it much easier for cybercriminals to acquire and use these tools [32]. RaaS offers ready-made ransomware tools and infrastructure in return for a share of the ransom payments. The accessibility and effectiveness of RaaS have contributed to another increase in ransomware incidents worldwide.

This research focuses on Windows ransomware, where three families are considered: Conti, WannaCry and Ryuk.

Conti

Conti ransomware emerged around 2020 and is believed to have been spread and developed by a Russian-based group named Wizard Spider. Conti is known for two things: its double extortion tactic, which encrypts the data and threatens to publish it if the ransom is not paid, and its breakneck encryption speed. Conti is believed to come from the Ryuk ransomware, developed by the same group, and was continuously updated and refined over the years, which made it one of the most persistent and adaptable ransomware families. Conti is said to be able to infect every version of the Windows operating system. The Conti ransomware operates as a RaaS model, allowing everyone to use the tools in return for a share of the ransom payments [6].

The malware uses a custom AES-256 implementation for encryption and up to 32 threads, which makes it stand out from other ransomware in terms of encryption speed. A simplified representation of how Conti works can be found in Figure 2.1. In addition to encrypting the data on the machine, Conti also deletes the shadow copies to prevent backups from being restored. Upon deployment, Conti terminates multiple services and tries to uninstall and disable Microsoft Defender and other monitoring tools. Conti targets files in all directories on all local and networked drives but excludes DLLs, executable files, system files, and link files.

WannaCry

The WannaCry ransomware emerged in May 2017 and is one of the most infamous ransomware attacks in history. WannaCry is believed to have been developed by the North Korean hacker group the Lazarus Group. WannaCry made use of the EternalBlue vulnerability in the Windows operating system. This exploit was created by the United States National Security Agency (NSA) and was leaked by the hacker group Shadow Brokers. This exploit allowed WannaCry to propagate across networks and infect more machines without requiring any user interaction. Once a machine was infected, all the files were encrypted, and a ransom note was displayed asking for a ransom in Bitcoin to decrypt the data.

The WannaCry attacks started on May 12, 2017, but they were drastically slowed a few hours after Marcus Hutchins registered a kill switch. The ransomware worked by contacting a domain (iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com) upon execution. If this domain is not registered and does not respond, it encrypts and infects the machine. So, after registering

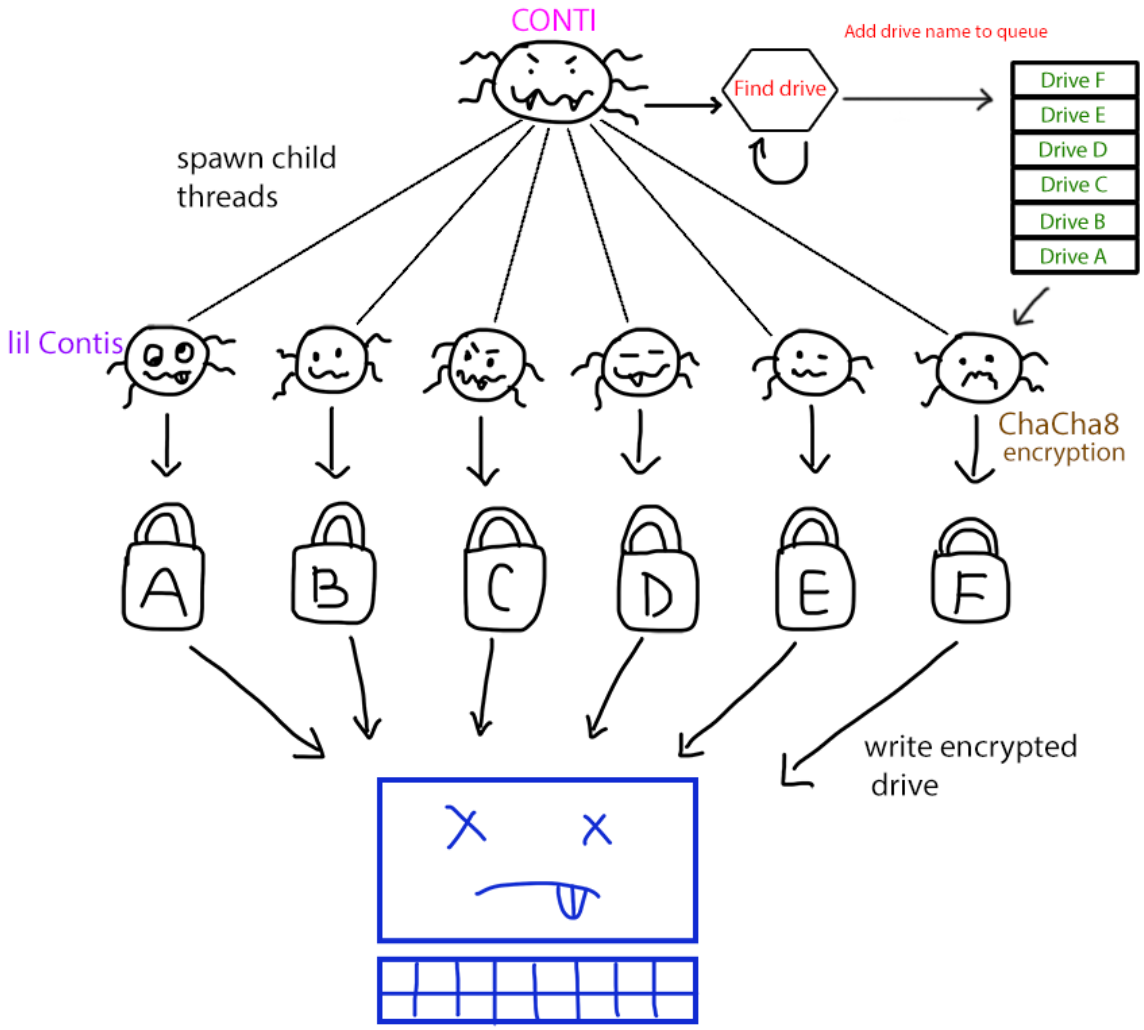


Figure 2.1: Visualisation of the Conti ransomware by Chuong Dong [15]

this kill switch, the malware stopped spreading as it stopped encrypting and infecting more machines.

Ryuk

Ryuk ransomware, first identified in 2018, is known for its highly targeted attacks on large organisations and critical infrastructure. It is believed to have been created by the Russian-based hacker group Wizard Spider and, unlike other malware, is not deployed openly but only on carefully planned operations. Ryuk uses Trickbot and Emotet to get an initial foothold in the system, after which reconnaissance is performed and privileges are elevated. Finally, files and data are encrypted.

Ryuk not only encrypts the data on the device but also tries to encrypt data on the rest of the network using a combination of RSA and AES encryption. For Ryuk, it is usually the case that multiple days or even weeks pass from the first sign of entry until the encryption starts, as it is a highly targeted attack. Ryuk also disables Windows's system restore function and deletes shadow copies to prevent backups from being restored [18]. Furthermore, the ransom payments demanded by the Ryuk ransomware are much higher than those of other ransomware,

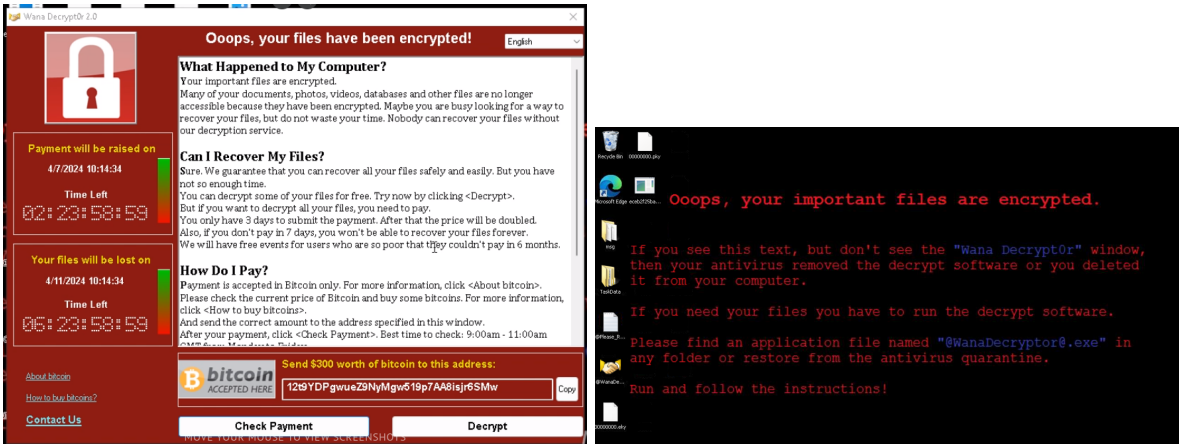


Figure 2.2: WannaCry ransom screen and the background it sets when the files have been encrypted.

which once more underlines the strategy to target prominent and wealthy organisations specifically.

2.1.2. Virus

Computer viruses are malicious programs designed to replicate themselves and spread from one machine to another. The idea of a self-replicating program first came up in the 1940s, and although some programs were created before, the first actual virus was created in 1986 by the Alvi brothers [41]. Their virus, called ”Brain,” infected the boot sector of a floppy disk with a copy of itself. The actual boot sector was then moved to another sector.

Today, computer viruses spread through various means, such as email attachments, software downloads or malicious websites. Once the virus is downloaded and executed, it does what it was made for: It starts replicating and spreading to other machines via network connectivity, shared files or removable media such as USB sticks. Even though ransomware and viruses seem very similar, their objectives still differ. Ransomware focuses on extortion by encrypting files and asking for a ransom to regain access. In contrast, viruses replicate themselves and cause damage by corrupting data and sometimes completely stopping system functionality.

2.1.3. Worm

Similarly to computer viruses, computer worms replicate themselves and spread onto other systems, but they do not need a host program. A virus writes its code to the host program, and once the code runs, the written virus program is executed first, whereas a worm does not need a host program but can run as an independent program or code. The first computer worm was the ”Creeper” worm in 1971 by Bob Thomas [38]. Creeper was mainly used to demonstrate the possibility of a computer program that could replicate itself and spread to other computers. It did not cause harm but displayed the message ”I’m the creeper: catch me if you can!”. This also led to the first antivirus program, Reaper, created to stop and delete Creeper.

Computer worms still pose a significant cybersecurity threat today, as they spread autonomously by exploiting existing vulnerabilities without user interaction. Modern worms, such as Stuxnet in 2010, have demonstrated that they can cause substantial damage. Stuxnet targeted SCADA systems and PLCs and had as a goal to target and disrupt Iran’s nuclear program by attacking Iranian nuclear facilities. Stuxnet used a chain of 4 zero-day exploits, which shows this worm’s high level of sophistication.

2.1.4. Root Kit

The term rootkit originally comes from a collection of tools Unix system administrators used to maintain root access to systems. However, over time, malicious actors began to exploit these tools to gain unauthorised access to systems and conceal their presence. Since then, rootkits have evolved into sophisticated malware designed to evade detection and establish persistent access to compromised systems. To this day, rootkits remain a significant threat, and precise estimates of damages caused by rootkits are challenging to quantify due to their stealthy nature.

Rootkits are often used as components of advanced persistent threats (APTs) targeting governments, enterprises, and critical infrastructure. Rootkits typically infect systems through various ways, including malicious email attachments, software downloads, and exploits targeting vulnerabilities in operating systems or applications. Once a system is infected, rootkits often employ stealth techniques to evade detection. They can hide their presence by manipulating system calls and using anti-detection mechanisms to mask their activities from security tools. They often include components such as backdoors, remote access Trojans (RATs), keyloggers, and spyware, enabling attackers to maintain control over compromised systems, steal sensitive information, and carry out malicious activities over an extended period.

2.1.5. Other

Many other malware types exist and have very different use cases, such as trojans, keyloggers, and spyware. A computer trojan, for example, has its name from the famous trojan horse. Trojans trick the victim into installing them by disguising themselves as legitimate and harmless programs, such as games or other utility programs. Once installed, they can perform various malicious activities, such as stealing sensitive information, granting remote access to the machine (Remote Access Trojan or RAT), or even installing additional malicious software.

Keyloggers, on the other hand, are a type of surveillance software designed to record and log keystrokes made by the victim. By retaining everything that has been typed, sensitive information such as passwords or personal information can be stolen. Keyloggers usually get installed as trojans or part of other programs and run silently in the background. They then communicate with a remote server, where they send all of the collected data.

Like keyloggers, spyware collects information about the user by collecting data such as browser history, program usage, and more. Like keyloggers, spyware mostly comes in the form of trojans disguised as legitimate programs or as part of another program. The data collected from the spyware is then sent to a remote server, where it is either sold or used for malicious purposes.

2.2. Malware analysis

Different methods exist to analyse malware and check how it behaves and what it does exactly. The two most common ones are dynamic and static malware analysis. Both function very differently and can be used in different scenarios. The following two sections explain what they are and their benefits and drawbacks.

2.2.1. Dynamic malware analysis

Dynamic malware analysis involves executing and observing a potentially malicious program in a controlled environment to study its real-time behaviour. This process typically occurs in a sandbox, a virtualised and isolated environment that mimics a real system but prevents the malware from causing any actual harm or escaping and spreading to other machines. During dynamic malware analysis, various system activities can be monitored during execution, such

as file modifications, network traffic, registry changes, process creation, and memory usage. Specialised tools and software are used to capture these interactions or are even built into these sandboxes, providing detailed insights into the malware's operational characteristics and identifying its payload and propagation methods. The sandbox used during this research is the ANY.RUNtD interactive sandbox [2], which makes it super easy to choose an operating system, configure it and run the dynamic analysis on any file or URL within seconds.

Dynamic malware, also known as behavioural analysis, allows us to study malware's actions and interactions directly to understand its functionalities and attack vectors. Using behavioural analysis, we can understand how malware spreads, whether it communicates with a C&C server, and how it causes harm while trying to stay hidden. By developing signatures and heuristics, we can even automate this process to a certain point and check if a program triggers specific rules by executing particular actions.

The main benefits of dynamic malware analysis are that it can be automated to analyse the malware for specific behaviour and that there is no need to access the application's code to see what it does. By running a malicious file, we can more easily verify that it is a true positive than just looking at the code. Furthermore, with dynamic malware analysis, we can detect unknown threats, such as zero days, as we can see what specific actions lead to and spot an exploit that would not be possible using static analysis.

Dynamic malware analysis, however, also comes with some drawbacks, mainly if the malware authors build in some detection tools to make this analysis harder. Some malware is sandbox-aware or virtual-machine-aware, meaning they detect if not run natively on an operating system. If this malware detects that they are in a sandbox, they might not show any malicious behaviour, making it impossible to analyse the malware using this technique. Another drawback of dynamic analysis is the possibility of malware escaping the sandbox if poorly configured. If malware manages to escape the sandbox or virtual machine, it may infect the actual system and compromise the network.

2.2.2. Static malware analysis

Static malware analysis involves examining the code and structure of a malicious file without executing it. This method includes disassembling or decompiling the malware to inspect its code and using various tools to extract and analyse features such as strings, libraries, functions, and metadata. Analysts look for indicators of compromise (IOCs), such as hardcoded IP addresses, domain names, and suspicious patterns. By examining the code, they can understand the malware's capabilities, intended behaviour, and potential impact. Tools commonly used in static analysis include disassemblers and decompilers like IDA Pro [24] or Ghidra [21].

In contrast to dynamic malware analysis, the malware is not run but reverse-engineered during static analysis. As the malware is not executed, there is no need for a live environment or sandbox, but caution should still be exercised. The aim of static analysis is similar to dynamic analysis, even though there may be some minor differences. It is essential for developing malware detection signatures, as the static features of the malware can be used to create unique fingerprints to recognise the malware in the future. Static analysis makes it possible to understand the underlying mechanisms and algorithms used by the malware. This method is beneficial for forensic investigations, where detailed code analysis can provide insights into how an attack was carried out and identify the artefacts left by the malware. Furthermore, static analysis is often a preliminary step before dynamic analysis, helping to identify specific areas of interest that would benefit from a closer inspection during runtime.

The main advantage of static analysis is that it is much quicker to extract specific features

than during dynamic analysis. Using automated tools and commands, metadata, strings, and malware structure can be extracted quickly. When using static analysis and signature-based detection, it is pretty simple to detect known malware, as these signatures are a unique fingerprint identifying a certain malware. Examining the code makes it possible to see specific functionality hidden during dynamic analysis or if the samples do not run while checking for its behaviour.

However, the advantages of static analysis do not come without any drawbacks, as some malware is also aware if a debugger is attached to the program. If the program detects a debugger, it might not execute any malicious behaviour and seem like a regular program. Another significant disadvantage is obfuscation and encryption. If the malware authors use custom packers and obfuscate the malware's code, then it becomes much more challenging to reverse engineer the code. Furthermore, not everything malware can do can be seen during static analysis. For example, if the malware downloads additional files from a remote server, these files cannot be found during static analysis. Furthermore, communications in general with a Command-and-Control server or similar cannot be seen with static analysis, as those will only be sent when the program is executed.

2.3. Executable files (.exe)

An executable file, commonly denoted with the ".exe" extension, is a file format used by Windows operating systems to run programs. These files contain binary machine code that the computer's processor can execute directly. Unlike text files or documents, executable files perform specified tasks or operations when run by the system and do not simply store data. They are essential for launching applications, performing system operations, and running utilities on Windows-based systems [11].

Executable files are created by writing source code in a high-level programming language, such as C++ or C#, and then compiling this code using a compiler. During this process, the compiler translates the high-level code into machine code by creating an object file (.obj). The object files are then linked using a linker, which combines these object files and any required libraries into a single executable file. It also resolves symbol references, such as function calls, and handles address relocation, ensuring the executable can run correctly in memory. Finally, the linker produces the final .exe file, which includes the executable machine code, metadata and all other necessary data to run the executable.

When an executable file is run, the operating system loads it into memory and begins execution. The Windows loader reads the executable file's header to determine how much memory to allocate for the program and where to place different segments, such as code, data, and resources. The file format used by Windows executables is the Portable Executable (PE) format, which contains headers and sections that describe the executable's layout in memory. The executable includes an import table that lists external functions and libraries on which the program depends. The loader resolves these dependencies by locating the necessary libraries (usually dynamic-link libraries or DLLs) and linking the executable's code to these libraries. Finally, once all sections are loaded and dependencies resolved, the loader transfers control to the program's entry point, as specified in the PE header. This entry point is the starting function (often main or WinMain) where the program begins execution. The CPU executes the machine instructions, performing operations defined by the program. During runtime, the program may request additional system resources, interact with hardware, and perform input/output operations through system calls. When the program completes its tasks, it terminates, releasing any allocated resources. The operating system then cleans up the process's memory space and handles any remaining background operations related to the

program.

As this research focuses on Windows ransomware, the data collected were only executable files that could be run on their own. This made analysis of these ransomware families much easier, as there was no need to compile anything or ensure that external dependencies were met. As an executable file is meant to be run without additional files or data, collecting the executable file was enough for the analysis.

2.4. MITRE ATT&CK

The MITRE ATT&CK matrix [33] is a comprehensive framework that categorises and describes cyber adversaries' actions and methods to achieve their objectives across various stages of an attack life cycle. It is structured around tactics, which represent the adversaries' goals at different stages, such as gaining initial access or exfiltrating data, and techniques, which are the specific methods employed to accomplish these goals, such as phishing or credential dumping. The matrix is organised into different categories to address distinct environments: Enterprise, which focuses on typical IT infrastructures; Mobile, which covers tactics and techniques relevant to mobile devices; and ICS (Industrial Control Systems), which pertains to the operational technology used in critical infrastructure sectors. This structured approach enables organisations to understand potential threats better, develop robust defences, and effectively respond to security incidents by mapping their security measures against known adversary behaviours.

The MITRE ATT&CK framework comprises 14 tactics for the Enterprise and Mobile categories and 12 for the ICS (Industrial Control Systems) category. The Enterprise matrix includes over 200 distinct techniques, while the Mobile and ICS matrices each encompass approximately 100 techniques. These numbers reflect the extensive range of strategies and methods adversaries might employ across different environments, underscoring the detailed and comprehensive nature of the MITRE ATT&CK framework. This research makes use of the MITRE framework to understand the goals of the malware authors as well as the behaviour of the malware itself. By looking at the differences in the behaviour of the samples, we can see how and why the malware authors employ specific techniques and how this affects the success rate of their programs.

2.5. Related work

Various methods and papers exist about all kinds of malware-related topics. This literature review will outline some of the most promising and helpful ones for this research. It should be mentioned that there are (almost) no papers that actively compare different generations of the same malware family, making it a gap in cybersecurity research. The main topics related to this research are malware detection, classification, evasion techniques, and dynamic malware analysis.

Malware detection

Detecting malware becomes more challenging as malware becomes more sophisticated and complex, so various research exists in this area. Javaheri et al. [27] propose a method to unpack and extract necessary features from malware and then generate a dataset with different mutations of this malware. This dataset can then be used to train a machine learning model to detect these rare classes of malware with a detection rate of 94-96%. Arabo et al. [3] propose a system that can detect ransomware by checking which DLL APIs are called, how many resources the program uses, and which files are opened by which processes. Using a trained machine learning model, they can detect zero-day malware and warn the user about

the infection. Aslan et al. [5] did a review about the most state-of-the-art malware detection and the problems associated with it. Most, if not all, of the methods described in the papers rely on machine learning algorithms, proving that anomaly detection works best if there is a well-trained model. The paper splits the methods into eight different categories and lists which specific technique can be used to detect that category.

Malware classification

One of the main problems in malware research is gathering data. Most properly labelled and complete databases are usually unavailable or hidden behind an NDA or license agreement. Most open datasets can contain many samples, but one can never be assured that those samples are correctly tagged and not false positives. Samuel Kim [28] devised an idea to use Windows Defender to classify the malware samples, as Virustotal was not feasible due to the limits on the maximum uploads per minute and day. The same setup can be used in this research to verify if the malware samples are not false positives and to extract initial information about them. Other papers, such as the ones by Prajapati et al. [35] or by Han et al. [23], transform the malware into images, then train machine learning models on them to classify them. Different generations of the same family might share similarities so that they can be found in the generated images. Other research might also rely on feature extraction to classify malware to its family, such as the paper by Liu et al. [29]. As before, most of the research is based on machine learning, not manual inspection.

Debugger and virtualisation detection

To analyse malware in the most secure way possible, the analysis is usually conducted in a sandboxed environment. However, malware authors sometimes build in measures to stop the malware from running when it can detect that a debugger is running or if it is inside a virtual machine. Salvatore Bova [10] wrote about how effective these evasion techniques are by applying these known evasion techniques and trying to bypass known anti-evasion techniques. Afianian et al. [1] wrote a survey about malware authors' different existing evasion techniques. They mention debuggers, file-less malware, and sandboxes and compare the various methods to each other. If the malware detects that it is running in a virtual machine or if a debugger is present, it might still do certain things. Still, all of its malicious behaviour might be completely disabled to avoid detection and make analysts' lives much harder.

Static vs dynamic analysis

There are two types of software analysis: static and dynamic. Both can analyse malicious software, each with its benefits and drawbacks. In contrast to dynamic analysis, the malware is not run at all but decompiled and reverse-engineered during static analysis. When analysing malware using dynamic analysis, the malware is run, and using a debugger or logs, it can be traced what the malware does. Even though both methods have their pros and cons, Moser et al. [34] talk about the limits of static analysis of malware. The paper discusses obfuscation techniques such as opaque constants, which make it possible to obfuscate the location where data is stored on the stack or in memory. Furthermore, they talk about binary transformation, meaning they change the binary without changing its functionality so that the same malware can have multiple forms. Or et al. [31] wrote a survey about state-of-the-art dynamic malware analysis. In their survey, they talk about everything from malware types to different techniques malware might use to evade detection, as well as tools that can be used to perform dynamic malware analysis.

3

Methodology

3.1. Safety Precautions

Working with live malware samples poses a high risk of data loss and infection of the machine and network. To mitigate these risks, it is crucial to establish an environment that isolates the machine and network, minimising the possibility of local infection. This research used a Digital Ocean droplet to store and analyse the malware samples. This droplet operated on Ubuntu 22.04, had 16GB of RAM, 4 Intel vCPUs, 320GB of storage and was connected to an entirely external network. Digital Ocean approved conducting malware research on their droplet.

As the droplet itself was running Linux, a Virtual Machine running Windows 11 was set up within the droplet environment. The connection to the droplet was made through the Remote Desktop Protocol (RDP), and live samples were never downloaded or transferred to the local machine or any machine connected to the local network. This fully isolated machine and network configuration significantly reduced the risks of infection, spreading, and data loss. This setup provided a secure and controlled environment for conducting malware research safely.

3.2. Data Collection

We first need to collect samples and build a dataset to analyse malware. The primary sources from which we get these samples are Malware Bazaar [30], and VX-Underground [40]. Both of these databases can be accessed via the web, and since there is no API to search for and download the samples, a Python script was used to scrape the sites.

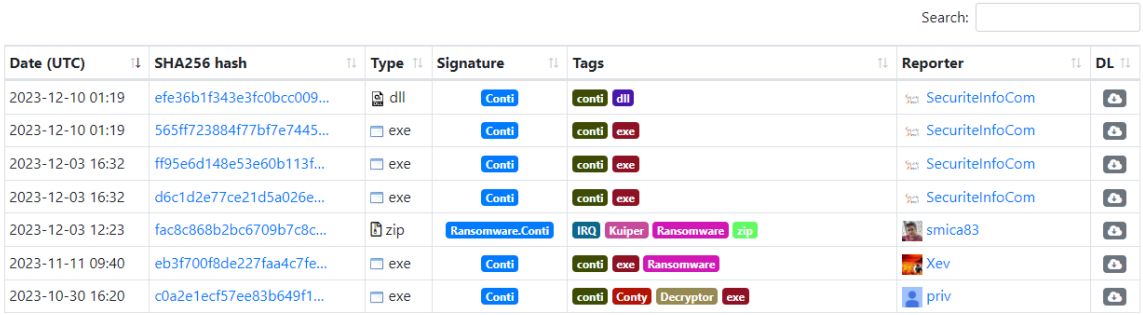
3.2.1. Malware Bazaar

Malware Bazaar has a querying system which allows us to search for different keywords. Their samples are stored with varying tags, such as the timestamp when it was uploaded and who uploaded it, the unique SHA256 hash and a signature, as can be seen in Figure 3.1. To filter the results by family, the 'signature' tag was used in the search, such that we looked for 'signature:<family name>', where the actual name of the malware family replaced the family name. We can filter and select specific samples and download them using the link provided for every sample using these results. As we are especially interested in standalone executable files, the type was filtered by 'exe' and other files were discarded.

The Python script extracts this information by querying the site using the search query and then writing all the results to a .csv file. All the fields extracted from the actual site are also

written to the .csv file, enabling the possibility of filtering later. Next, we use the data from the .csv file and filter the results by type, only keeping the .exe files. We then used the download link for each sample and downloaded the malware. It should be noted that, as explained in a previous section, these samples were not downloaded to a local machine but to an isolated droplet.

For the Conti family, we downloaded 150 samples from Malware Bazaar; for the Ryuk family, 40 samples; and for the WannaCry family, 960 samples. We did not use all of the samples, as some were tagged wrong or for other reasons, so these numbers are a bit lower for the final samples that were kept. Additionally, to ”disarm” the files, they are not stored as executables but are compressed and archived into .zip files, protected with the password ’infected’. This adds an extra layer of protection as downloading and accidentally clicking the executable becomes impossible. Furthermore, as Malware Bazaar is an open database where everyone can upload samples, not every sample tagged as Conti, for example, is guaranteed to be a sample from the Conti family. So, to confirm that we have true positives, we had to do some verification, as explained in the following section.


Search:

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2023-12-10 01:19	efe36b1f343e3fc0bcc009...	dll	Conti	conti dll	SecuriteInfoCom	
2023-12-10 01:19	565ff723884f77bf7e7445...	exe	Conti	conti exe	SecuriteInfoCom	
2023-12-03 16:32	ff95e6d148e53e60b113f...	exe	Conti	conti exe	SecuriteInfoCom	
2023-12-03 16:32	d6c1d2e77ce21d5a026e...	exe	Conti	conti exe	SecuriteInfoCom	
2023-12-03 12:23	fac8c868b2bc6709b7c8c...	zip	Ransomware.Conti	IRQ Kuiper Ransomware zip	smica83	
2023-11-11 09:40	eb3f700f8de227faa4c7fe...	exe	Conti	conti exe Ransomware	Xev	
2023-10-30 16:20	c0a2e1ecf57ee83b649f1...	exe	Conti	conti Conty Decryptor exe	priv	

Figure 3.1: Results for the query ’signature:Conti’ on Malware Bazaar.

3.2.2. VX-Underground

Similarly to Malware Bazaar, VX-Underground does not offer a direct API to query and download samples filtered by a specific family. This means that we also scraped the site and automated the download process. Contrary to Malware Bazaar, VX-Underground does not have a direct querying system. Still, their files are ordered in a well-structured file system, as seen in Figure 3.2. By traversing to the directories through ’Home/Samples/Families’, we can find a list of different malware families and pick the ones we are interested in.

Once again, we used a Python script to extract all the download links from the website and downloaded them to the droplet. VX-Underground is not an open database, meaning not everyone can submit samples, making the samples likely to be true positives. However, this would also, in this case, still need to be verified afterwards to ensure that the dataset that we create only contains samples from the given family and that all these samples are also executed. Also, contrary to Malware Bazaar, VX-Underground only offers .exe files and not .dll or others, making the filtering obsolete in this case.

For the three families we were interested in, we collected 183 samples for the Conti family, 14 for the Ryuk family, and one for WannaCry. Similarly to Malware Bazaar, to ”disarm” the samples so they do not get executed by accident when downloading them, they are archived and compressed into a .7z file protected with the password ’infected’. The actual executable file can then be found when extracting the archive. Finally, with the combined samples, we collected enough for each family to analyse them and draw meaningful conclusions.

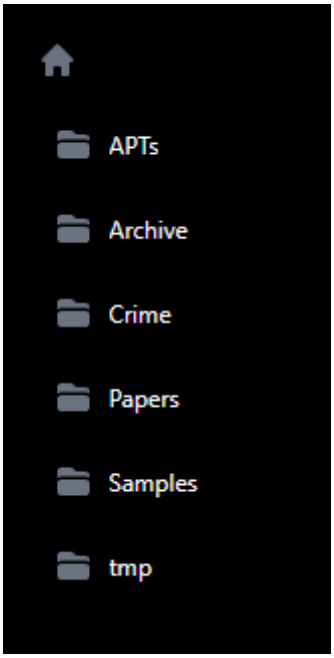


Figure 3.2: The file structure of the VX-Underground database.

3.2.3. Virustotal

As we want to get as much information about a sample as possible, we analyse the samples using dynamic malware analysis using the AnyRun interactive sandbox. We also used the data from Virustotal, which offers various API endpoints that can be queried and return different results. As we are mainly interested in general information about the file, the file behaviour and sandbox details, we decided only to query 3 different endpoints, namely:

- api/v3/files/sha256Hash, which gives general information about the file, such as dates of when it was created or first found, information about imports and headers, etc.
- api/v3/files/sha256Hash/behaviour_summary, which gives a summary of its behaviour, such as commands it executes, processes and registry keys
- api/v3/files/sha256Hash/behaviour_mitre_trees, which gives the verdict of different sandboxes as well as MITRE tactics and techniques

All three endpoints return data split into different fields. Not all of the fields are directly useful for this research, so not all of the fields were kept; only the important ones were parsed and extracted. The Data Analysis section explains which fields exist, were extracted, and were used.

3.3. Data Verification

3.3.1. Classification with Windows Defender Antivirus

To ensure that all of the downloaded samples are also true positives, we have to do some data verification. For this purpose, we use Microsoft Defender Antivirus, as this program not only classifies the samples by family but also gives a variant name to it, making it extremely useful for later data analysis when comparing different versions of the same family. The way that Microsoft is naming malware can be seen in a blog post [25]. Microsoft assigns each file a type to explain what it does, such as Trojan or Ransom, followed by the platform being the compatible operating system. Next, they add the family and variant based on common

characteristics and behaviour. The variants increase in chronological order, meaning variant A comes before variant AD, which in turn comes before DA. To add some additional information to the classification, they also add a suffix. An example of such a classification can be found in Figure 3.3.

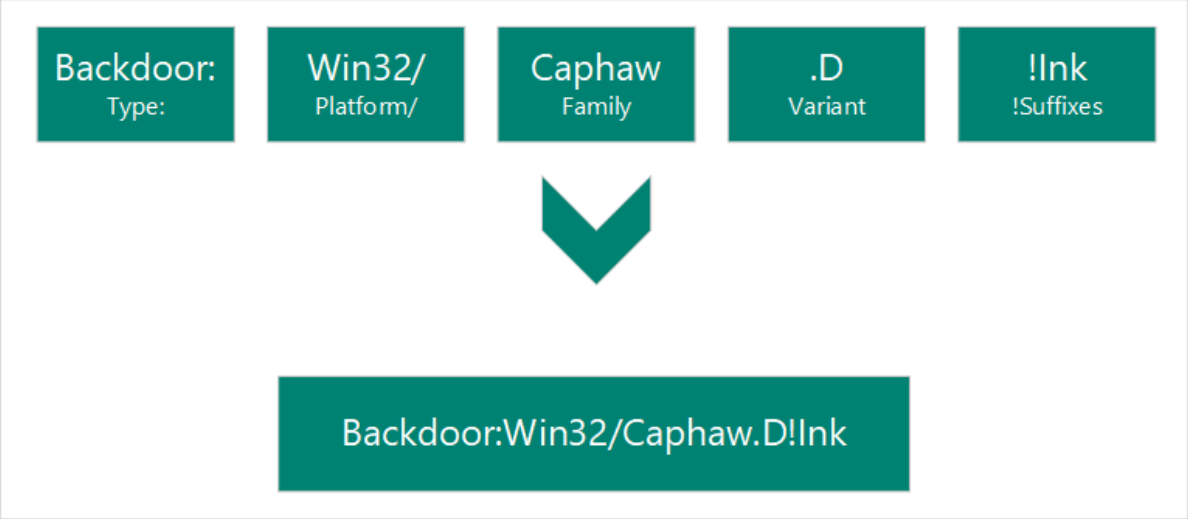


Figure 3.3: Microsoft's naming scheme for malware. [25]

All downloaded malware samples were transferred to the Windows virtual machine on the droplet. With the Defender turned on, all the samples were extracted, and most were immediately blocked and deleted by the Defender. Since Defender logs all its actions, we can extract all its classifications and remove possible false positives. The logs which we are interested in can be found in "C:/ProgramData/Microsoft/Windows Defender/Support/MPDetection-<timestamp>.log" on the Windows machine, where <timestamp> is a timestamp from when the file was created. Sometimes, it took a while for the logs to be updated; in this case, it helped to run a quick scan on the machine, as this forces an update of the log file as Defender scans the whole system.

By using and automating this method, we could quickly get a classification of the samples we collected and remove any false positives. After this initial analysis and classification, we ruled out multiple samples we collected. For Conti, we now had 222 samples left, excluding false positives and duplicates. When repeating the same for the other two families, we had 38 samples left for Ryuk and 108 for WannaCry. Approximately one-third of the samples we collected for each malware family proved to be false positives.

It's important to note that since there is no complete documentation for Microsoft Defender Antivirus and the program is not open-source, some suffixes and names cannot be explained. For example, we assume that Conti, CONTI and ContiCrypt are the same family, as we also see similar behaviour from all of these samples. Furthermore, we did not find any explanations for the suffixes !pz, !MTB and !MSR. We asked these questions on the Microsoft forums [13] [12] but have not received any response, which has helped to clear this up. So, for the remainder of this research, we assumed it was the same family and disregarded the suffixes and their meaning.

The following subsections summarise the final true positives collected during the data collection.

Conti

The samples which were left were grouped into several variants, more precisely 24 variants for Conti, as depicted in Figure 3.4, namely:

- Ransom:Win32/CONTI
- Ransom:Win32/CONTI.DA!MTB
- Ransom:Win32/CONTI.DC!MTB
- Ransom:Win32/Conti!pz
- Ransom:Win32/Conti.A!MTB
- Ransom:Win32/Conti.AD
- Ransom:Win32/Conti.AD!MTB
- Ransom:Win64/Conti.IIP!MTB
- Ransom:Win32/Conti.IPA!MTB
- Ransom:Win32/Conti.MAK!MTB
- Ransom:Win32/Conti.PA!MTB
- Ransom:Win32/Conti.PAA!MTB
- Ransom:Win32/Conti.PLD!MTB
- Ransom:Win32/Conti.SW!MSR
- Ransom:Win32/Conti.WEN!MTB
- Ransom:Win32/Conti.ZA
- Ransom:Win32/Conti.ZC
- Ransom:Win32/Conti.ZD
- Ransom:Win32/Conti.ZE
- Ransom:Win32/ContiCrypt!MTB
- Ransom:Win32/ContiCrypt.MFP!MTB
- Ransom:Win32/ContiCrypt.PADD!MTB
- Ransom:Win32/ContiCrypt.PB!MTB
- Ransom:Win64/ContiCrypt.PF!MTB

For some variants, more samples could be found than for others. Still, a single sample of a different variant should give some insight into behavioural differences.

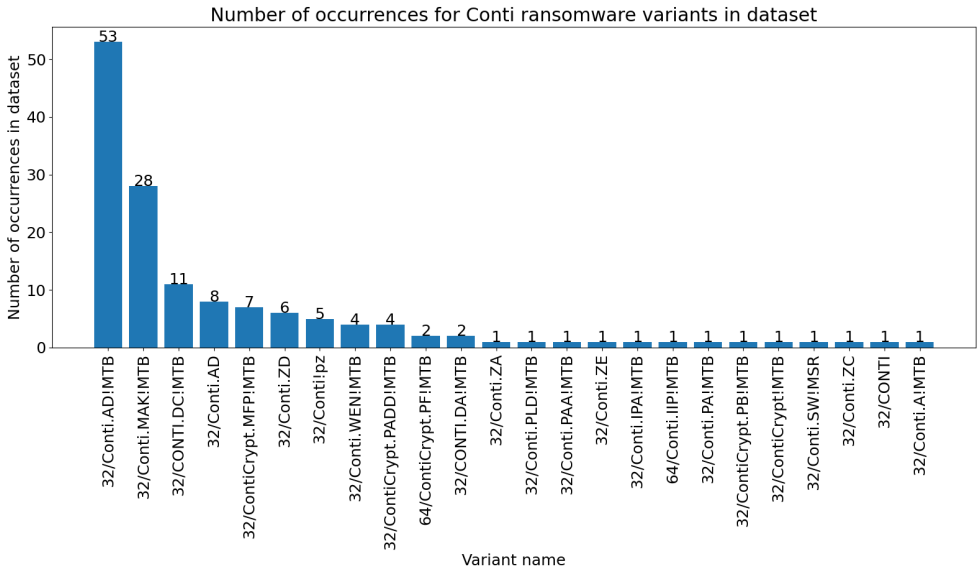


Figure 3.4: Different variants of Conti with the number of samples in the dataset.

WannaCry

In the end, we had four variants for the WannaCry ransomware, namely the following ones:

- Ransom:Win32/WannaCrypt
- Ransom:Win32/WannaCrypt!pz

- Ransom:Win32/WannaCrypt.G
- Ransom:Win32/WannaCrypt.H

The number of samples we have for every variant can be seen in Figure 3.5.

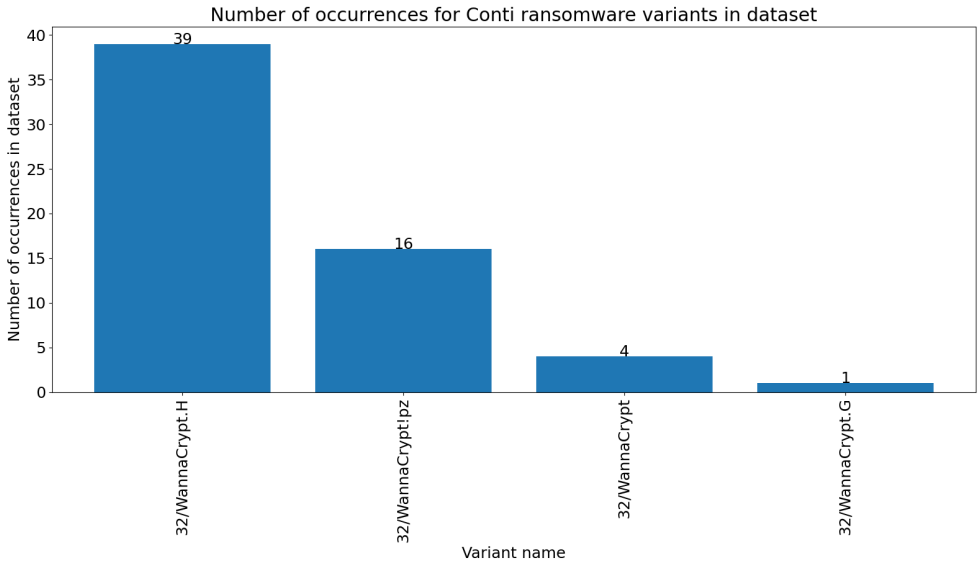


Figure 3.5: Different variants of WannaCry with the number of samples in the dataset.

Ryuk

For Ryuk, we had the least amount of samples overall between the three families, but we still managed to find five different variants, namely:

- Ransom:Win32/Ryuk.AA
- Ransom:Win32/Ryuk.B
- Ransom:Win64/Ryuk.PA!MTB
- Ransom:Win64/Ryuk.PB!MTB
- Ransom:Win32/Ryuk.SB!MSR

As before, the number of samples we found for each variant can be seen in Figure 3.6.

3.3.2. Manual verification

When analysing the samples using the data extracted from Virustotal or running them in the AnyRun sandbox, some inconsistencies could be observed for multiple samples. Due to this, we decided to manually analyse every single sample to verify that, even though we already had the classification from Microsoft Defender Antivirus, the samples did indeed belong to the said family. We loaded the samples into the AnyRun interactive sandbox and analysed the file's behaviour.

For some samples, we found that it was ransomware, as they encrypted files, but it was a completely different ransomware family. The most prominent families we saw instead of Conti were, but are not limited to, BlueSky, PUTIN, Blackhunt or Monti ransomware. We explain this as they are all clones, heavily inspired by the Conti ransomware or even thought to be developed by the same group. They all use threading for encryption and the same tactics and

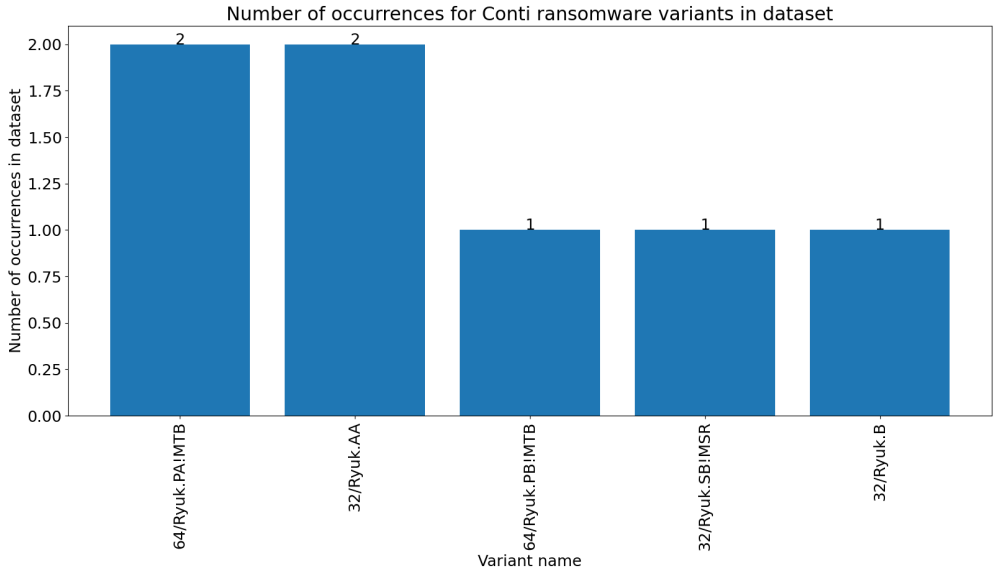


Figure 3.6: Different variants of Ryuk with the number of samples in the dataset.

techniques in the rest of their behaviour. Overall, 28 samples proved to be a different ransomware than Conti. This clearly shows some flaws with the pure signature-based detection system of Antivirus software.

Other samples were not recognised as Windows executables, even though they had the .exe extension and seemed to be valid executable files. They did not change anything on the system, as they did not run at all. Other samples were corrupted and only started the *WerFault.exe* program, which is the Windows error reporting tool. This hints at a broken executable, which could be due to stripped headers or similar actions, which can completely disarm a malicious executable file, as it cannot be properly loaded anymore. As we are interested in comparing the behaviour of the collected samples on different versions of the same OS, we decided also to discard those files.

Finally, some samples started, but nothing ever happened. Some of these samples ran for over 10 minutes without doing anything; no processes were spawned or created, no files were modified, and no other machine settings were touched. Other samples started but were terminated after a bit without doing anything. When looking at reports from different sandboxes and Antivirus scanners, it turned out that these were most likely also not Conti samples but Trojans or other malware, which might be sandbox-aware and did not execute. We also decided to discard those samples.

Finally, after repeating this process for every sample of every family we collected, we ended up with 143 true positives for Conti, where we can confidently say that they are indeed Conti samples. We also eliminated multiple WannaCry and Ryuk samples and ended up with 60 samples for WannaCry and 7 for Ryuk. As Conti is the main family we are analysing, we have enough samples from different variants to get meaningful results. For WannaCry, the number of samples is also big enough, especially as there are not as many variants within the samples we collected. Finally, even though we only have seven samples and five different variants for the Ryuk family, we can still draw conclusions from the data, especially when comparing the different families. Conti inherits most of its features from Ryuk, so we predict many similarities.

It should be noted that even though almost all of the WannaCry samples did not execute, this is due to how WannaCry works. WannaCry contacts a specific kill-switch domain (*iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com*), and if this domain does not respond, it continues with encrypting the files and tries to spread to other machines on the network. As this domain was registered when WannaCry came out in 2017, all of the samples are sink-holed by the domain and do not execute anymore. However, as this is expected behaviour from the samples, we classified them as true positives and kept them in the dataset.

3.4. Data Analysis

We analyse the data using AnyRun, mostly to see if we can see certain differences between different versions of the same operating system. We use Virustotal to check mainly for differences in the behaviour of the samples. If necessary, we can supplement this information with the additional insights we receive from the analysis on AnyRun.

As mentioned before, the data retrieved from Virustotal was retrieved from their API and split into various fields. A complete list of the objects and information that was retrieved can be found on their respective API documentation pages [19] [20] [7]. A lot of this data is not present for many samples, is a duplicate from another field or does not add any helpful information to this research. Due to this reason, only a smaller selection of fields was extracted and further analysed. A list of extracted fields can be found in Appendix A. Most of these fields were further split into sub-fields for which only the essential and helpful information was extracted.

After an initial look at the extracted data, it became apparent quickly that the selection could be further narrowed down. Specific fields had no data, or insufficient samples had this data to draw conclusions, so the fields were discarded. Furthermore, some fields contained the same information in a different form but did not add anything new if one of those fields was already used, hence only a single field was used and the others were discarded. Finally, extracting useful information from some fields was not feasible, as they had too much information or, in a way, could not be used. One example of the latter is the file field. Given that every sandbox uses a different operating system configuration, mostly with random files, the information in this field is quite cluttered and without a lot of pre-processing, extracting any data is infeasible. Plus, the same information could also be retrieved from other fields.

Eventually, the final list of fields which were used for this research was:

- **creation_date** → The creation or compilation date of the executable
- **first_seen_itw_date** → The first time an Antivirus scanner detected this specific sample in the wild
- **first_submission_date** → The first time this sample was submitted for analysis on Virustotal
- **pe_info**, more specifically the **import_list** sub-field → Shows which libraries are most often used by malware
- **sandbox_verdicts**, more specifically the **category** and **malware_classification** sub-fields → Gives an insight into how the same variant can be classified differently
- **text_decoded** → Shows the evolution of the ransom notes
- **tags** → Shows most crucial file behaviour
- **signature_matches** → Shows which rules were triggered by the malware, which in turn shows the behaviour of the malware

- **sandboxes** → Provides MITRE tactics and techniques detected by the various sandboxes on Virustotal. With these, we can see what malware does and how it tries to achieve its goals

The analysis conducted on AnyRun had more of a manual practice than automating it. The main point of the AnyRun analysis was to check if the malware behaves differently on different versions of the same operating system. For example, if a sample encrypts the files on Windows 11 but does not encrypt anything on Windows 7, then it could show that it is specific to that version of the OS or is using an exploit related to the version of the operating system.

Overall, many Python scripts were written to fetch, parse, and analyse the data retrieved from both sources. Most of the steps were automated as much as possible, but some parts were still manually worked on. These scripts can be found in the GitHub repository [22].

4

Evaluation

This chapter shows the different results and conclusions from the collected data. The results are split into various sections according to what exactly we compared. The first section contains general information about the samples followed by an analysis per variant. Afterwards, we look at the samples over time and continue with a broader comparison of all three families. Finally, the last section discusses the results by comparing the same samples across multiple versions of the same operating system.

These results are compared by looking at the behaviour of the samples. With the data we collected from Virustotal and ANY.RUN, we look at the samples in more detail to see their similarities and differences. The general, variant-based, time-based, and environment-based comparisons are all applied to the Conti family, and only the family-based comparison uses all three families. All the results and scripts can be found in the GitHub repository of this research [22].

4.1. General

A general impression from the Conti family is that it was around for about two years before it slowly disappeared, and no more new samples were created or released. In Figure 4.1, we can see that Conti's main lifespan was from early 2020 to the beginning of 2022. The graph shows three different timestamps: the first time the sample was detected in the wild by an Antivirus scanner, the first time it was submitted to Virustotal and the date when it was created or compiled. As the first seen in the wild time is more representative of the actual date of the sample, we plot that one instead of the first submission. However, not every sample has the first seen in the wild timestamp, so we plot the first submission time instead. The creation time is plotted for every sample where it is present.

The central hypothesis at the start was that the core of each sample from the same family would be the same, and only small behavioural changes could be seen, which was confirmed after the analysis. Looking more closely at the different fields we extracted, we can see that almost all samples are built similarly to achieve similar results. Still, we can also see some more minor differences.

When looking at the imports of the files, there is only a single library used by every executable, kernel32.dll. This makes sense, as this library is used to expose most of the Win32 base APIs to applications. The functions used range from memory management to input/output (I/O) operations, processing, and thread creation. Within this library, mainly the same functions are

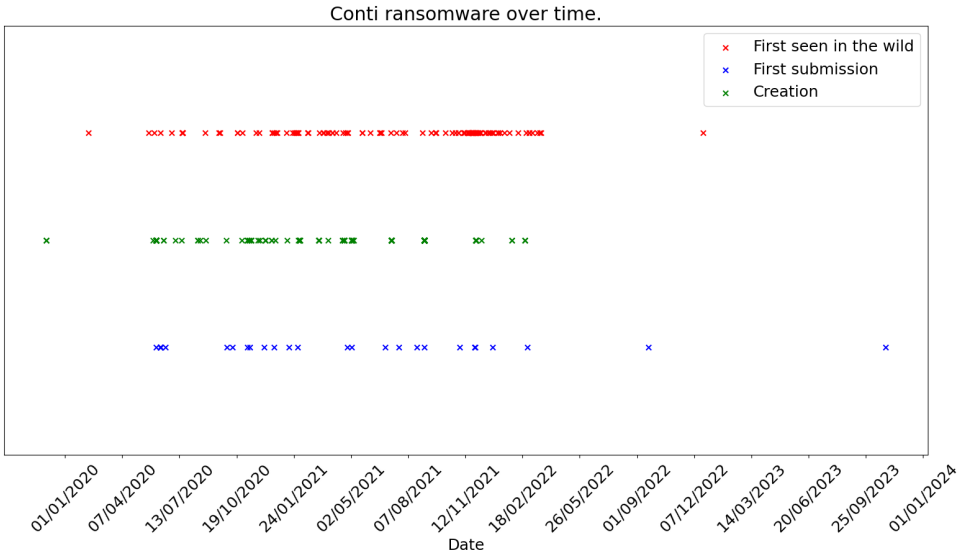


Figure 4.1: Conti ransomware over time, depicted in first seen in the wild, first submission, and creation dates.

used by about 80-90% of the samples, which all have something to do with processes and acquiring a handle for specific files and threads. However, a lot of functions are only called by less than 20% of the samples, clearly showing different behaviour for a minority of them. Some functions are even only called by a single sample. The functions in question are primarily used to free and exit threads, create mutexes, sleep or format strings in a certain way.

The other most common DLLs are user32.dll and shlwapi.dll. These libraries are primarily used for input processing and working with file paths or registry operations. Not all samples use these libraries, but only 117 out of 143 or 98 out of 143, respectively. This is not unexpected as these libraries also contain some of the most crucial behaviour from ransomware. The other libraries, however, show the differences between the samples, as they are used by less than 25% of the samples, with the majority only being used by two or only one sample. A complete list of the libraries and functions can be found in the GitHub repository. All the libraries and the number of times the Conti samples have loaded them can be found in Table B.1.

When looking at the classifications of the sandboxes from Virustotal, one would expect that all existing antivirus scanners detect known samples, but this is not the case. Furthermore, tagging malware based on what it does also provides different results. Virustotal offers multiple sandboxes through which the sample is run and extracts the verdicts of these sandboxes on whether the sample is malicious, harmless, or undetected. Of over 448 verdicts spread across the 143 samples, 335 were classified as malicious, which means that around 75% of the verdicts were correct, as all of the samples have been proven to be true positives. From the remaining verdicts, only ten classified the samples as harmless, and 103 could not detect them and classified them as undetected. With every classification, the sandboxes also tag the samples with what they think the sample is. From 143 total samples, the tag MALWARE was given to 97 samples, TROJAN to 73 samples and the RANSOM tag was only given to 50 samples, so less than 50%. There were also some rare classifications, such as CLEAN with three samples, SPREADER and EVADER with two and UNKNOWN_VERDICT with 25 samples. Some classifications also did not give any tag at all.

Moving on to the ransom notes, we have nine different ransom notes for the Conti family. Some of the ransom notes only differ from the email addresses that were used, but we can see one specific ransom note, which came up more often than others, around 50% of the time, namely:

```
All of your files are currently encrypted by CONTI strain.
As you know (if you don't - just "google it"), all of the data that has been
    encrypted by our software cannot be recovered by any means without
    contacting our team directly.
If you try to use any additional recovery software - the files might be
    damaged, so if you are willing to try - try it on the data of the lowest
    value.
To make sure that we REALLY CAN get your data back - we offer you to decrypt 2
     random files completely free of charge.
You can contact our team directly for further instructions through our website
     :
TOR VERSION :
(you should download and install TOR browser first hxxps://torproject.org)
hxxp://m232fdxbfmbrcehbrj5iayknxnggf6niqfj6x4iedrgtab4qupzjlaid.onion
hxxpS VERSION :
hxxps://contirecovery.best
YOU SHOULD BE AWARE!
Just in case, if you try to ignore us. We've downloaded a pack of your
    internal data and are ready to publish it on out news website if you do not
     respond. So it will be better for both sides if you contact us as soon as
    possible.
---BEGIN ID
-
---END ID---
```

When looking at the actual behaviour of the samples, we can look at the tags and signatures which they matched. From 16 tags, only five were appointed to more than 50% of the samples. 109 samples loaded runtime modules, 108 had long sleeps in their behaviour, probably for evading security measures, 93 directly accessed the CPU clock, 87 had active measures against static analysis and had a debug environment detection built-in, and 79 samples called the WMI (Windows Management Instrumentation). We can gain a more detailed insight into what the samples do from the signatures, as it has to show particular behaviour to trigger these rules. We see quite a difference in behaviour from the signatures from the samples, as the most present signature is barely seen within 50% of all the collected samples. Here, it is essential to note that malware might not directly execute a command, for example, but tries to find a workaround so as not to trigger too many rules simultaneously and to be as covert as possible. The main signatures show behaviour that we would expect from ransomware, such as obfuscated strings on the stack, which can be the ransom note, links to the C&C server, or references to a public RSA key, which can be used for the encryption of the files. However, the main findings that point to ransomware are triggered when something is being encrypted, such as ’encrypt data using Salsa20 or ChaCha’, ’encrypt data using XOR’ or ’create or open file’. Since we know that the same family has different variants, we can also see some outliers in the signatures, such as ’packed with generic packer’ or ’packed with UPX’, which were only present for a single sample. This clearly shows that specific behaviour is not present for all the samples, and even if the majority uses the same behaviour, slight but pretty notable differences still exist.

Finally, we can also look at the tactics ransomware uses and the techniques it applies to reach its goals. In short, the tactics are what the malware authors try to do, and the techniques are how they try to achieve it. MITRE defines 14 enterprise tactics, which are again split into

different techniques. We found 12 out of the 14 existing enterprise tactics for the collected Conti samples. For each tactic, we also recorded the techniques which were applied. These results can be found in Table 4.1. When looking at the specific techniques, we can see wildly varying numbers. TA0001 does not apply in the sense of how the malware lands on the machine, as during the testing, the executable was manually loaded on the machine, but from the techniques, we can see that from the machine, it's scanning for open hosts on the network. This could later be used to spread to more machines. For TA0002, the malware authors use shared modules to execute their malicious payloads. They load these shared modules as executables into processes to have reusable code. Other techniques that can be seen are using the COM (Component Object Model) or the native Windows API. TA0003 is about getting a persistent foothold in the system, mainly achieved by changing or adding registry keys and adding processes that boot on startup or side-loading DLLs. This tactic is, however, not widely used within Conti, as the main goal is encrypting the data, so there does not necessarily need to be a persistent foothold for later use. For the following tactic, TA0004, not many techniques exist for the number of times the tactic was detected. But mainly, the malware is trying to use processes to execute code instead of executing it by itself to find a workaround for getting higher privileges. With TA0005, we see the correlation with the tags and signatures, as almost every sample tries to evade security measures. Conti is not looking for credentials with TA0006, but it tries to discover the whole system and network before proceeding. By looking for system information, processes, file directories or even virtualisation and sandboxes, it tries to map out the entire system to see what it can encrypt and how it can spread and infect other devices. Additionally, when a system is infected, the malware tries to spread laterally. Conti delivers payloads through shared storage locations or other file transfers. Finally, the remaining tactics, such as TA0040, clearly show that it is ransomware, as the malware deletes backups, stops services used for recovery and encrypts files.

Conti shows the expected behaviour of ransomware, and we also see some differences between samples from a general point of view. The following sections will break up this analysis into more detailed parts to make the actual comparison more accessible and transparent.

4.2. Variant

When analysing the samples from a variant point of view, we expect the similarities to be held within the same variant. The outlying behaviour we observed in the general comparison is predicted to be isolated into different variants. Overall, we expect the samples to differ more from one to another if the variants are further away from each other, meaning if we compare variant DA to variant DC, we expect to see fewer differences than when we compare DA to ZD, for example, as Defender classifies the samples in a chronological naming order.

The first fields we examine are the dates, which enable us to create a timeline of when the different variants were released. Once again, we split the analysis into three distinct fields: the first time the sample was seen in the wild, the first submission date, and the creation date. We expect the first seen-in-the-wild dates to be the most representative, as even if a sample has been around for a while, it does not necessarily mean it was also submitted to Virustotal immediately.

When looking at the first seen in the wild dates in Figure 4.2, we can see that most of the variants emerged around the same time in the middle of 2020. Furthermore, we can also see that some variants are around longer than others, for example, the CONTI.DA!MTB variant was actively deployed from the middle of 2020 to the beginning of 2022, with one new sample even being detected at the end of 2023. Due to the limited number of samples for certain variants, we cannot draw specific conclusions from them. Still, we can reason that since fewer

Table 4.1: MITRE tactics found in the 143 Conti samples.

Tactic	Name	Count	Description	Techniques
TA0001	Initial Access	13	The adversary is trying to get into your network.	2
TA0002	Execution	85	The adversary is trying to run malicious code.	10
TA0003	Persistence	47	The adversary is trying to maintain their foothold.	9
TA0004	Privilege Escalation	54	The adversary is trying to gain higher-level permissions.	10
TA0005	Defense Evasion	121	The adversary is trying to avoid being detected.	28
TA0006	Credential Access	8	The adversary is trying to steal account names and passwords.	1
TA0007	Discovery	117	The adversary is trying to figure out your environment.	20
TA0008	Lateral Movement	85	The adversary is trying to move through your environment.	4
TA0009	Collection	33	The adversary is trying to gather data of interest to their goal.	3
TA0011	Command and Control	50	The adversary is trying to communicate with compromised systems to control them.	7
TA0040	Impact	78	The adversary is trying to manipulate, interrupt, or destroy your systems and data.	6
TA0043	Reconnaissance	1	The adversary is trying to gather information they can use to plan future operations.	1

Note: The 'count' column contains the number of samples which use this tactic, and the 'techniques' column contains the number of different techniques that are used for this tactic.

samples are available for one particular variant, it was perhaps not as successful as others. Hence, it was only used for a short period, whereas others were used over a more extended period, which is also why more samples exist.

Starting from the imports, the kernel32.dll does not give much information, as it is used by every sample and, hence, also by every variant. By looking at the outliers, namely the libraries only used by a single sample or two samples, we can see that they are all from a variant where we only collected a single sample. Namely variants "ContiCrypt.PB!MTB" and "CONTI". When looking into these two variants in more detail and at the functions they use from the libraries, we can see that the main functions are still related to input, path parsing or threading. The way the "CONTI" variant uses some libraries is interesting, as it seems to use different libraries to use functions such as ord() to hide stack strings. This might be a way to hide from antivirus scanners and still be able to do the same behaviour, but it is unknown to previous signatures. Another interesting finding is that libraries that are not present in many samples are still present in different variants. For example, the "oleaut32.dll" library is used by six samples, which are classified as four different variants, namely "Conti.IIP!MTB", "CONTI", "Conti.MAK!MTB", and "Conti.AD!MTB". This is especially interesting, as our initial hypothesis was that similar variants from the naming would be similar in their behaviour. However, this clearly shows

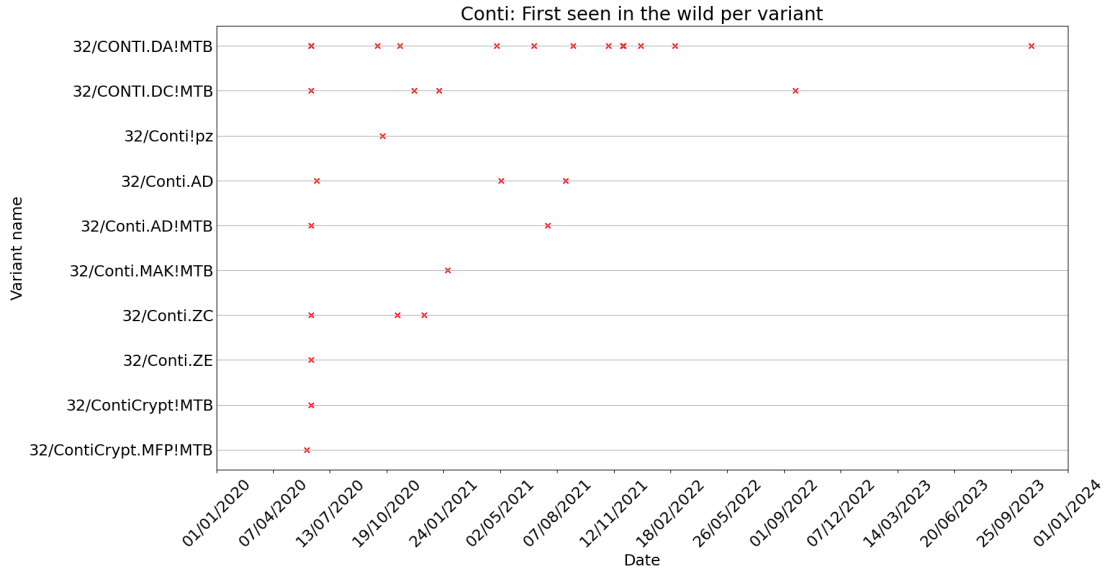


Figure 4.2: First seen in the wild dates for Conti ransomware split by variant.

that entirely different variants can have very specific information that links them to unrelated variants. Another example is advapi32.dll, which is present in 10 samples but six different variants or ole32.dll, which is present in 11 samples and seven variants. Furthermore, when we look more into the variants, we can also see differences, such as Conti.AD!MTB, which has 53 samples in the dataset, but only 14 use the ws2_32.dll library, or only three use the ole32.dll library. So, even within the same variant, we still see differences from the imports, which does not necessarily mean that there is a difference in behaviour, as the same variant could have evolved and used different libraries to execute functions, which led to the same result. Table B.1 shows all of the DLLs the Conti samples use with a description and count of how many samples use them.

The sandbox verdicts show similar results on a variant basis than they did on the general analysis. The first result that stands out is that one variant is more often classified as harmless than others. There is, at best, a single sample classified as harmless for all the variants. In contrast, the rest are classified as malicious or undetected, but variant ”ContiCrypt.MFP!MTB” has only seven samples but is classified as harmless six times out of 35 classifications. This shows some apparent evasion from antivirus scanners to trick them into thinking they are legitimate programs. The interesting part about this is that this was not one of the variants that used different libraries for the same behaviour, as this variant did not stand out at all from the analysis of the imports. For the other variants, it’s usually a clear classification of them being malicious, as anything between 75%-100% classifications are malicious. One interesting variant concerning this fact is ”Conti.MAK!MTB”, as only 50% of all the classifications are malicious, and for the rest, the sample remained undetected. This is a far better ratio than all the other variants. The only other variant coming close to the same ratio is ”ContiCrypt.MFP!MTB”, with a 65% ratio of being classified as malicious. The tags do not show any conclusive patterns, as only a single variant, ”Conti.ZD”, was classified only as RANSOM. All the other variants have received multiple different tags, with the only other interesting one being ”Conti.ZA”. However, this might be related to the sample size, as only a single sample is classified as this variant in the whole dataset. So, overall, we can see that some samples are better at avoiding malicious

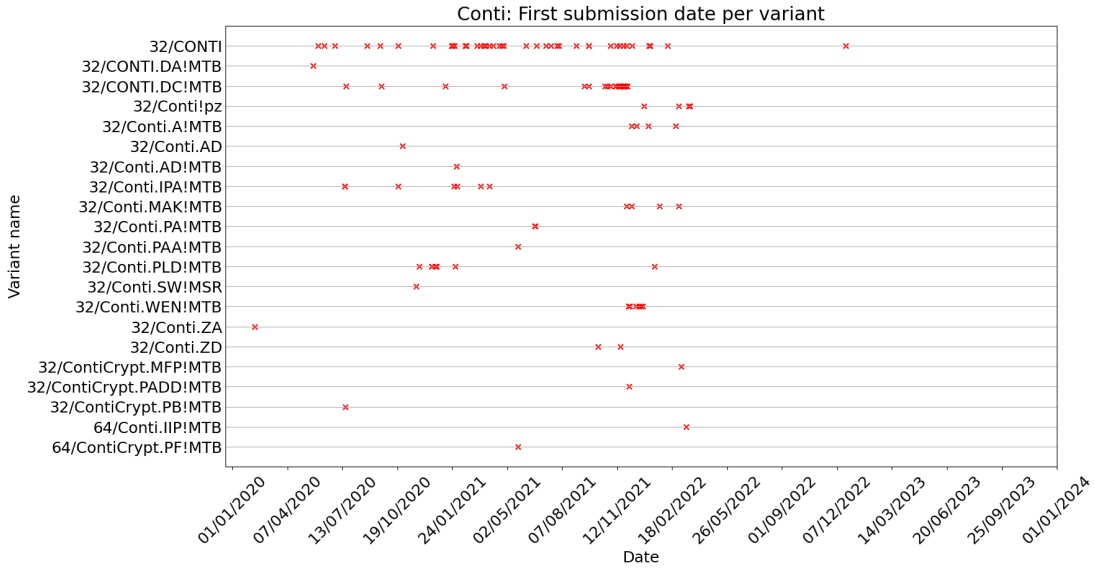


Figure 4.3: First submission dates for Conti ransomware split by variant.

classification than others. Still, other than that, we do not learn much more about the variants as the tags are not very conclusive but rather general, with tags such as MALWARE, TROJAN, and RANSOM being used for most variants.

The ransom notes are rather unvaried, as most variants only use a single ransom note. The only interesting variant in this category is ”Conti.AD!MTB”, which has seven different ransom notes. All the other variants either only have the same ransom note for all the samples, or they are the same ransom note with a slight difference in them, which can be only a few words which changed. The hypothesis here is that we can see an evolution of the ransom notes over time. Still, since some variants were present over a more extended period, this might explain the different ransom notes we can see for ”Conti.AD!MTB”, for example, as this variant was around for two years. For other variants, for which there are also multiple samples, the period in which they were active is smaller, and hence, we can also see less difference in the ransom notes. One example is the ”Conti.MAK!MTB” variant, for which the dataset contains 28 samples. However, this variant was only active for around three months, which is why we only see a single ransom note.

From the tags alone, we cannot see much difference in the behaviour of the different variants. The tags present in most of the samples can also be seen in almost all of the variants, but when we look at other less common tags, we can see that those occur in variants with fewer samples. For example, the PERSISTENCE tag, which was only allocated for a single sample, can be found in the ”ContiCrypt.PADD!MTB” variant, for which we only have four samples, and the tag cannot be found in any other variant. The PERSISTENCE tag is especially interesting, as Conti is not known for being a manually executed ransomware but rather doing the encryption autonomously without manual interception from the malicious actor. The PERSISTENCE tag could hint at a variant that installs a backdoor or another way to make it possible for the malware authors to return to the machine. Another interesting tag is the IDLE tag, which we could connect with the LONG_SLEEPS tag, but as it is classified differently, there is a difference in behaviour. For sleep calls, the malware is specifically told to wait until executing the following command, whereas idling means the program is not doing anything. The IDLE tag is

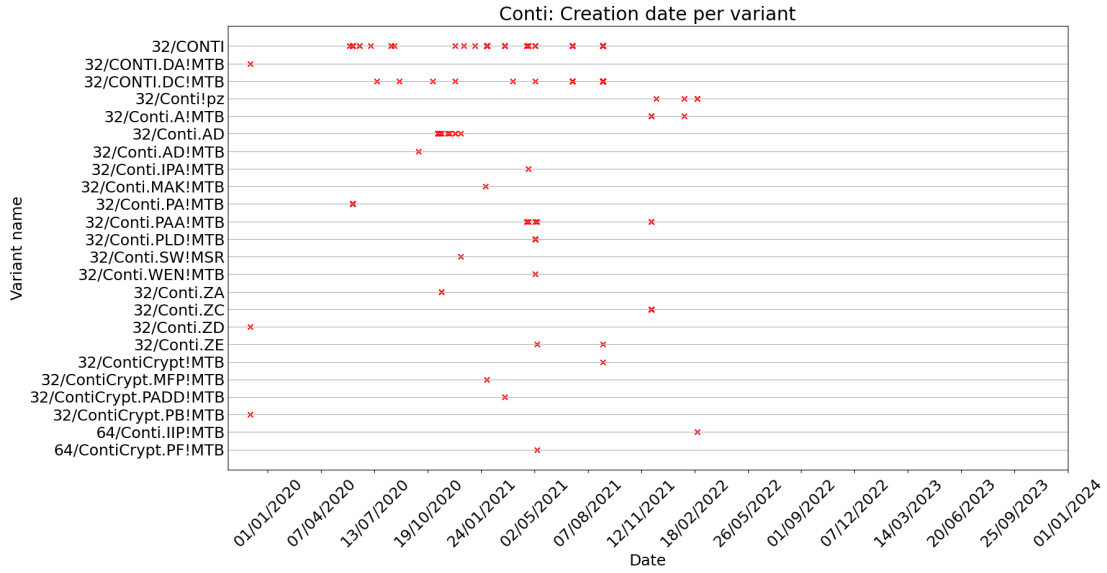


Figure 4.4: Creation dates for Conti ransomware split by variant.

also only present for three variants, namely ”Conti.AD!MTB”, ”Conti.AD”, and ”Conti.PA!MTB”. Here, we have two very similar variants, which show that the changes are perhaps related to the classification of the variants from Microsoft Defender Antivirus.

When looking more into the behaviour of the files, specifically at the signatures the samples matched, we can see that the samples show differences in their behaviour even within the same variant. The best example is the most common variant in the dataset, ”Conti.AD!MTB”, which has 53 samples and 47 different triggered signatures. The interesting part, however, is that many of these signatures are only matched by a single sample. 18 out of 47 signatures are only matched by a single sample, and 27 match less than ten samples. This means that over 50% of the samples show behaviour that cannot be found in other samples, even though they were classified as the same variant. Once more, we can only speculate why this happens, as the documentation of the Defender classification is limited, and the program is not open-source. When looking more specifically at the individual behaviours, we can see that only 14 of 24 variants use obfuscated stack strings to hide the malicious behaviour from security measures. Even though all of the samples have been checked to verify that they do encrypt the files on the machines, not all of them trigger a signature which detects file encryption. When looking through the different variants, only around 50% trigger a signature which has something to do with encryption, hashing or encoding. The signatures in question are: ”hash data using murmur2/3”, ”hash data with CRC32”, ”encrypt data using Salsa20 or ChaCha”, ”Encrypts content of user files”, ”encrypt data using RC4 PRGA”, ”encode data using XOR”, ”encode data using Base64”. The numbers we see could have something to do with how they go about hiding their behaviour, but one other important fact is that for eight variants, we have no signatures which were triggered at all. These are variants where we have a single sample, such as ”Conti.ZA”, but also variants which have up to 28 samples in the dataset, like ”Conti.MAK!MTB”.

Finally, we’ll take a look at the tactics and techniques from the different variants in order to see in detail how they achieve a particular goal and if this behaviour differs from one variant to another. Table 4.2 gives an overview of how many samples from each variant use a specific

tactic. As in the general overview, we can also clearly see that certain tactics are used more than others. With the overview of the specific variants, we can now also see that even if a lot of samples do not use a tactic, the samples are still part of different variants and not all from a single cluster. The best example for this is tactic TA0001, which is only used by 13 samples but is present in 6 different variants. The techniques in this tactic do not differ at all from the different variants, however, as all of them use either the ”Exploit Public-Facing Application” or ”External Remote Services” techniques. Tactic TA0002 is present in 16 out of the 24 variants in the dataset. From the ten techniques that are used by the samples in the dataset, some rely on external files or modules, whereas others use the built-in Windows functionality, such as the WMI or the native API. The most common technique for tactic TA0002, however, is T1129 ”Shared Modules”, which is used by every variant which uses TA0002 and also by more than 90% of the samples which use said tactic. TA0003 uses a significant number of different techniques, which differ more from one another than the ones from TA0002, for example. There is no technique which is omnipresent for every variant, but we can mostly see either T1574.002 ”DLL Side-Loading” or T1547.001 ”Registry Run Keys / Startup Folder” for all variants. Less common techniques used to achieve persistence are, for example, T1176 ”Browser Extensions”, which are only used by 3 out of the 24 variants. TA0004 is relatively uneventful, as different variants use the same techniques to escalate their privileges, mainly T1055 ”Process Injection” and T1547.001 ”Registry Run Keys / Startup Folder”. TA0005 is interesting, as most of the samples use this tactic, but also because there are by far the most different techniques. Even if a variant only has a single sample, then that sample uses multiple techniques to evade the defences to stay hidden. Similarly to TA0004, TA0005 also most commonly uses T1055 ”Process Injection” as a technique to evade defences. For TA0007, some variants are more thorough when it comes to discovery than others; for example, the variant ”Conti.PLD!MTB” uses 12 out of 20 different techniques, even though we only found a single sample. The variant ”Conti.ZA”, in contrast, only uses four techniques to discover the machine it is trying to infect. TA0008 once again has an interesting characteristic, as every variant which uses this tactic uses all four techniques which were found, or only T1080 ”Taint Shared Content”. Every variant except ”Conti.PLD!MTB” uses three of the four found techniques and does not use T1080. For T0011, we cannot see a clear pattern, but mostly a mix of T1095 - ”Non-Application Layer Protocols” or T1071 - ”Application Layer Protocol” proxies and encrypted channels. TA0040 shows clear signs of ransomware, as all 18 variants except one which uses this tactic use T1486 - ”Data Encrypted for Impact”, which shows that they are encrypting the files. Next to encrypting the data, the most common techniques for impact are T1499 - ”Service Stop” and T1490 - ”Inhibit System Recovery”, which both point to deleting backups or shutting down/removing recovery programs. The only variant which does Reconnaissance and uses TA0043 is ”CONTI.DC!MTB”. It is also interesting to note that for two variants, namely ”Conti.SW!MSR” and ”Conti.ZE” no tactics or techniques were found at all.

4.3. Over time

Malware changes over time based on which security measures are put in place due to the developments of the field. When looking at the same malware family over time, we can see that different variants emerge. Still, those variants do not necessarily come out one after the other, but some may also be developed at the same time, such that different variants emerge at the same moment in time. This section groups the samples based on their creation date and looks for similarities and differences to find a trend in the evolution of the ransomware in question. As the Conti family was active for about two years, we decided to use a time interval of 3 months each, as a smaller interval might result in too few samples for some time,

Table 4.2: Conti variants and the MITRE tactics they use

Tactic / Variant	01	02	03	04	05	06	07	08	09	11	40	43	Samples
CONTI	0	1	0	0	1	0	1	0	0	0	0	0	1
CONTI.DA!MTB	0	1	0	0	2	0	2	1	1	0	1	0	2
CONTI.DC!MTB	3	6	1	0	7	0	6	6	3	2	4	1	11
Conti!pz	1	5	0	2	5	0	5	3	4	4	3	0	5
Conti.A!MTB	0	1	1	1	1	0	1	0	0	1	0	0	1
Conti.AD	0	7	4	4	7	0	4	0	1	1	4	0	8
Conti.AD!MTB	5	44	23	27	52	7	52	35	17	23	29	0	53
Conti.IIP!MTB	0	1	1	1	1	0	1	1	0	1	1	0	1
Conti.IPA!MTB	0	0	0	0	1	0	1	1	0	0	1	0	1
Conti.MAK!MTB	0	0	1	1	16	0	16	14	0	6	9	0	28
Conti.PA!MTB	0	1	1	1	1	0	1	0	1	1	1	0	1
Conti.PAA!MTB	0	1	1	1	1	0	1	0	0	1	0	0	1
Conti.PLD!MTB	1	1	1	1	1	0	1	1	1	0	1	0	1
Conti.SW!MSR	0	0	0	0	0	0	0	0	0	0	0	0	1
Conti.WEN!MTB	1	3	1	2	4	1	4	4	2	1	4	0	4
Conti.ZA	0	0	0	0	1	0	1	1	0	0	1	0	1
Conti.ZC	0	1	0	0	1	0	1	1	0	0	1	0	1
Conti.ZD	0	0	0	0	5	0	5	5	0	0	5	0	6
Conti.ZE	0	0	0	0	0	0	0	0	0	0	0	0	1
ContiCrypt!MTB	0	1	1	1	1	0	1	1	0	1	1	0	1
ContiCrypt.MFP!MTB	0	7	6	6	6	0	6	6	0	5	6	0	7
ContiCrypt.PADD!MTB	2	4	3	3	4	0	4	3	3	1	4	0	4
ContiCrypt.PB!MTB	0	0	0	1	1	0	1	0	0	1	0	0	1
ContiCrypt.PF!MTB	0	0	2	2	2	0	2	2	0	1	2	0	2
Variants	6	16	14	15	22	2	22	16	9	15	18	1	

Note: Every cell represents the number of samples from a variant which uses the specific tactic. Every tactic should be prepended with ”TA00”. The last row represents the number of variants that use said tactic, and the last column represents the number of samples a variant has.

and a larger time interval might result in too many samples and too little detail. We chose the earliest creation date from all the samples as the starting date for our intervals and then just used increments of three months to group the rest of the samples. This way, we ended up with nine groups with varying amounts of samples per time interval. The exact number of samples and time intervals can be found in Table 4.3.

When looking at how the libraries were used over time, we can clearly see that it started with only very few libraries, and over time, more and more libraries were used. However, towards the end of Conti’s lifetime, the number of libraries which were used went down once more. In the first six months, the authors of Conti only used the kernel32.dll library, next to the shell32.dll and shlwapi.dll DLLs. The following six months show an increase in the libraries which are used, as from 23/11/2020 to 21/02/2021, 8 and 13 libraries were used, respectively. This trend, however, did not continue, as during the start of 2021, the number of libraries dropped once more to 3 before rising to 12 in the middle of 2021. Towards the end of Conti’s lifetime,

Table 4.3: Time intervals and number of samples for the Conti ransomware.

Time Interval	Samples
27/02/2020 - 27/05/2020	3
27/05/2020 - 25/08/2020	1
25/08/2020 - 23/11/2020	22
23/11/2020 - 21/02/2021	11
21/02/2021 - 22/05/2021	19
22/05/2021 - 20/08/2021	29
20/08/2021 - 18/11/2021	13
18/11/2021 - 16/02/2022	28
16/02/2022 - 17/05/2022	14

the number of libraries used dropped to 3 and stayed stable between 3 and 5. When looking more specifically at the functions which are called inside the libraries, we can see that the samples started very basic and only used functions to create and work with threads, load libraries, or execute commands. It was only after a year that the malware started to become more sophisticated, and different libraries were used for the same behaviour or completely new behaviour was added to the samples. Around August 2020, the malware started using ws2_32.dll, a library for network communications, as Conti began to upload files from the victims as they now moved over to the double extortion tactic. At this point, Conti did not only encrypt the files but also downloaded them and threatened to publish them if the ransom was not paid. At the end of 2020, Conti saw its first libraries that used GUI elements or other graphical functions. Afterwards, the behaviour of Conti seems to revert to its original form by only using basic libraries and not adding any additional behaviour; instead, it only uses simple libraries and functions that provide the main functionality. A year before the end of Conti's lifespan, the malware became more sophisticated again, using libraries for multimedia support, graphics device interfaces, and more. After three months, the samples that were created and released once again had only the basic libraries, and they did not change anymore for the last nine months of Conti's active deployment.

Regarding the detection and classification of the samples, we can clearly see that the early samples of Conti did not have any built-in evasion tactics, as all of them are classified as malicious with the tag MALWARE or RANSOM. Only after six months does the ransomware start to stay undetected from different sandboxes and even manages to be classified as harmless once. This trend continues over time, and Conti becomes even more evasive, as around 10-20% of all the samples during 2021-2022 remain undetected from sandboxes and antivirus scanners. It's only towards the end of its lifetime, in 2022, that this number drops significantly, and the malware was once more classified almost uniquely as malicious. We can explain this trend by constantly updating the databases with signatures from antivirus scanners. When malware is first released, it is not detected, but once a signature is added to the database, the samples are detected and blocked. With the constant updates Conti received, it managed to evade the security measures for a significant amount of time. Still, when support was finally dropped, the samples became ineffective as the security measures finally caught up with their evolution. The tags that the malware received are mostly the same, and it was only at the end of 2021 that the malware received different tags such as EVADER, SPREADER or CLEAN. For the other periods, it always received the tags MALWARE, RANSOM and TROJAN.

The ransom notes show a clear evolution in contrast to the variant comparison we did earlier. The ransom notes during the first nine months of Conti's operation all started with either ”Your system is LOCKED.”, ”The network is LOCKED”, or ”The system is LOCKED”. Multiple email addresses followed the ransom note to contact for the decryption key. It was only around the end of 2021 when the ransom notes started to become more detailed, mentioning that the files were encrypted using Conti ransomware and that it is not possible to decrypt the files without the key. This message mainly stayed the same until the end, and the only changes were an addition that backups were also deleted and that they can offer proof that they can decrypt files by making it possible to upload a few files and get them decrypted. During the variant analysis, we did not see a clear timeline, as the variants also range over a long period, which makes the evolution less visible.

The tags of the ransomware over time do not give us a good insight into the malware, as most of the tags are given to every single period. ”LONG_SLEEPS”, ”DIRECT_CPU_CLOCK_ACCESS”, and ”RUNTIME_MODULES” can be found in every single period. We can, however, see an evolution of the malware when looking at the tags, as six months after the first Conti sample, all the samples become debugger-aware and are tagged with ”DETECT_DEBUG_ENVIRONMENT”. It is also noticeable that initially, not all the samples in the same period were debugger-aware. Still, the more time goes on, the higher the percentage of samples which are aware of a debug environment. Furthermore, after six months of updates, the samples all used Windows Management Instrumentation to check for user input, possibly to steal confidential information such as passwords.

When looking at the signatures of the samples, we can see that during every single period, the malware contains obfuscated stack strings. After the initial six months of deployment, the malware authors made it harder for the files to be detected by signature-based detection methods by building in packers. The only period in which packers were used was from August 2020 until November 2020. Furthermore, as we have already seen when analysing the tags, from 2021 onwards, the malware has done more data collection by collecting information such as file sizes, disk information, and even geographical location. 2021 really marks the point at which Conti ransomware became especially aware of debugging tools and tries to evade defences as much as possible by executing actions such as delaying the execution or simply detecting antivirus software or similar. During every single period, except for the first three months, the specific encryption signature is also triggered, as the ransomware encrypts data using RC4 PRGA, Salsa20 or ChaCha. In the later versions, Conti also starts to reference public keys, possibly used for encryption and uses HMAC and modulo calculations in assembly. These features cannot be found in the early stages of the malware but clearly show the evolution and sophistication of the updated versions of Conti. Another exciting discovery is that Conti only seems to trigger signatures based on registry keys in 2 different periods and not in any other. The first time is approximately nine months after the first Conti sample emerged, and the final one is at the very end of Conti's lifespan.

Lastly, the tactics and techniques show us how the behaviour of the samples changed over time. Table 4.4 shows the tactics which the samples in a particular period use. In the comparison over time, we can clearly see a more even distribution when a specific tactic was applied than when the variant analysis was applied. The tactics are mainly used over Conti's whole lifespan, whereas with the variants, we saw some variants that used much fewer tactics overall. When diving deeper into the techniques Conti uses over time to accomplish the tactics, we can also see a trend that changes the longer Conti is around. TA0001 does not show any exciting behaviour, similar to the variant comparison. For TA0002, we can see a clear evolution of the techniques used over time. Initially, the only techniques used were the native

API, shared modules, and the Windows COM (Component Object Model). In the later periods, Conti started using the Windows service control manager and scripts to execute commands, and it was only in the second half that Conti started to use WMI for the majority of the samples. TA0003 shows that Conti uses registry keys for persistence at the start and the end of its lifetime, but it does not seem to use them in the middle. Overall, there is no discernible trend in the techniques used for persistence, as the malware authors constantly use different tricks to achieve their goal in this scenario. Privilege Escalation shows a clear path of using T1055 ”Process Injection” as the primary technique to gain administrator rights on the system. During every single period, the adversary tries to inject code into processes to avoid process-based defences. Next to process injection, we also commonly see T1543.003 ”Windows Service” as a technique. One thing which is noticeable in the overtime analysis is that in the later periods, generally, more techniques are applied for privilege escalation than in the beginning. For TA0005, we can see that defence evasion was always one of Conti’s main focus points. From the very start, Conti used at least five different techniques to evade the defences, and most of the techniques also persisted throughout its whole lifespan. The most common techniques Conti uses are, for example, T1562.001 - ”Disable or Modify Tools” or T1027 - ”Obfuscated Files or Information”. We can also confirm this behaviour from the tags and signatures we analysed earlier. More specific techniques which Conti also used only during particular periods are T1218 ”System Binary Proxy Execution”, which was only used during two periods at the very start, or T1070.004 ”File Deletion”, which was also only used during the first three periods. TA0006 was not used at all in the beginning but was later adopted to be a part of every period from 2021 onwards, with the same technique being used at all times. For the discovery, Conti became more and more aware of the system it was running on over time. Initially, it only checked the files, processes and software, but in the later periods, it also became aware of debuggers, virtualisation and systems connected to the network. So, TA0007 clearly shows that the malware becomes more sophisticated over time. TA0008 shows no changes over time, except at the end of 2021, where instead of using all four found techniques to move laterally, Conti only uses T1080, which delivers payloads to remote systems via shared storage. Conti started to collect data mainly by hijacking the browser sessions at the start of 2020 but completely dropped this approach at the beginning of 2021 for the rest of its lifetime. The following year, Conti mainly collected data from user input or local system sources. To establish command and control, Conti initially transferred the tools or other files it needed directly from an external source and only later dropped this approach to adapt multiple protocols. In the beginning, Conti used non-application layer protocols as well as application layer protocols, and, towards the end, it also started adapting proxies and encrypted channels. Overall, Conti proceeded to use a variety of ways to exfiltrate data, which can be seen throughout its lifetime. The impact of Conti, or TA0040, is, of course, T1486 for all of its lifespan, which encrypts the data on the machine on which it was deployed. Only around a year after its first deployment, Conti also started to stop or turn off services which could be used for recovery and started deleting backups and shadow copies. TA0043 can only be found on 21/02/2021-22/05/2021, and it uses a single technique during this period, namely T1595 - ”Active Scanning”.

4.4. Family

When comparing different families, it is essential not to get lost in the actual differences of the families. Even though all three families are ransomware, they are still very fundamentally different. A straightforward example is Conti using multi-threading when encrypting, and WannaCry is not doing this at all. This analysis tries to find some similarities, especially regarding general behaviour, by looking at MITRE tactics and techniques, as well as imports and matched signatures. We expect to see similar behaviour between Conti and Ryuk, as

Table 4.4: Conti samples over time and the MITRE tactics they use

Time period \ Tactic	01	02	03	04	05	06	07	08	09	11	40	43	Samples
27/02/2020 - 27/05/2020	0	3	3	3	3	0	3	0	1	3	1	0	3
27/05/2020 - 25/08/2020	0	1	1	1	1	0	1	0	0	0	0	0	1
25/08/2020 - 23/11/2020	0	18	17	17	21	1	18	0	4	5	4	0	22
23/11/2020 - 21/02/2021	1	6	0	0	8	0	7	4	2	1	3	0	11
21/02/2021 - 22/05/2021	6	12	3	3	15	2	15	15	9	7	13	1	19
22/05/2021 - 20/08/2021	2	21	7	10	27	1	27	22	8	17	21	0	29
20/08/2021 - 18/11/2021	0	4	0	2	11	2	11	11	2	7	3	0	13
18/11/2021 - 16/02/2022	0	10	9	10	19	1	19	18	1	7	17	0	28
16/02/2022 - 17/05/2022	2	7	4	5	13	1	13	13	4	3	13	0	14
Variants	4	9	7	8	9	6	9	6	8	8	8	1	

Note: Every cell represents the number of samples from a period which uses the specific tactic. Every tactic should be prepended with ”TA00”. The last row represents the number of periods during which the tactic was used, and the last column represents the number of samples a period contains.

the same ransomware group supposedly created Conti and differences in behaviour between Conti and WannaCry.

When looking at the samples in general, we saw that for Conti, different samples, classified as other variants, had differences in behaviour. In WannaCry, this is not at all the same, as almost all 60 samples seem to have precisely the same structure and seem to behave the same way. The numbers in the Conti samples varied quite a bit, from single samples showing the behaviour to a handful, half of the dataset or all of them, whereas for WannaCry, 57 of the 60 samples are almost identical based on the data which was collected. Even though we only collected seven true-positive samples from Ryuk, we still see similarities in the number of data compared to Conti, which seems to prove our initial hypothesis.

When looking more in detail into the imports between the samples in Table D.1 and Table D.2 in Appendix D, we see that out of 8 DLLs used in the WannaCry ransomware family, only two are not used by Conti, namely msvcp60.dll and msvcrt.dll. For Ryuk, way more DLLs are used, but Conti also uses the majority. There are 5 DLLs, which are used by Conti but not by Ryuk, and 21, which are used by Ryuk but not by Conti. In total, they have 11 DLLs in common. As Ryuk is very targeted and specialised ransomware, it makes sense that it uses many more DLLs than common ransomware, as the tools and exploits are most likely more sophisticated than for simple RaaS ransomware like Conti.

The sandbox verdicts also differ from one family to another, with the Conti family being the one with the least malicious classifications. 75% of all the classifications for the Conti samples were malicious, 23% were undetected and only 2% slipped past the detection and got classified as harmless. WannaCry was classified 99% of the time as malicious, and only a single classification was undetected. Ryuk had a malicious classification rate of 90%, and the other 10% were undetected. The tags that were assigned to the samples differ across all three families. WannaCry is almost exclusively classified as MALWARE and RANSOM, whereas Conti is also classified as TROJAN quite a lot, and Ryuk is classified as SPREADER, TROJAN and EVADER. Furthermore, Conti has the highest UNKNOWN_VERDICT classification out of all three families, which might have something to do with the large number of different samples

that are available for the Conti family.

Table 4.5: Tags received by sandboxes for the Conti, Ryuk and WannaCry samples in the dataset.

Tag	Conti	Ryuk	WannaCry
MALWARE	38.49%	38.46%	67.65%
RANSOM	19.84%	30.77%	25%
TROJAN	28.97%	7.69%	2.94%
CLEAN	1.19%	0%	0%
SPREADER	0.79%	15.38%	0%
EVADER	0.79%	7.69%	2.94%
UNKNOWN_VERDICT	9.92%	0%	1.47%

The tags and signatures from the families show some differences in behaviour as well. Every tag that is present in the Ryuk family is also present for the Conti family, except for the CHECKS_DISK_SPACE tag. Ryuk, however, has only received seven different tags, whereas Conti has received 16 different ones. WannaCry also received 16 different tags, of which six are not present in the Conti samples. WannaCry is more specifically checking for network adapters and USB storage devices, simply creating macros or deleting itself when it is done. Conti and Ryuk have 38 signatures in common from around 85 total each one has. WannaCry and Conti have approximately the same number of similar signatures which were triggered, namely 35 out of around 80-85 each. From the common signatures, 23 can be found in all three families. The signatures that can be found in all three families are basically only interactions with the Windows operating system and nothing peculiar to the ransomware itself. The signatures present in all three families are allocating memory, creating or opening files, enumerating PE sections or parsing the PE headers, changing registry values or terminating processes. Overall, we can see that the specific signatures are more specific to the actual families, and we do not see them in all the families. For example, WannaCry is hashing and linking more data than Conti or Ryuk and is also setting up more connections via different sockets. Ryuk, on the other hand, focuses more on privileges, checking for stealthy ways to stay hidden and does way more enumeration than the other two families. In general, Ryuk does way more enumeration on the whole system by checking for VMs, analysing the cursor movement and enumerating all the files on Windows. WannaCry, on the other hand, seems to gather information about the system and then proceeds to encrypt all of the files on the system.

Finally, when looking at the MITRE tactics and techniques of Conti, Ryuk and WannaCry, we can observe that Ryuk is using the least amount of different tactics, which comes as a surprise, as Ryuk is very targeted and seems to specialise in a lot of different areas. Table 4.6 show the different tactics which are used by the families as well as how many different techniques they apply to achieve said tactics. Overall, we can see that different families use different techniques in order to achieve similar tactics. Ryuk is the only family that does not try to obtain any initial access, which makes sense as Ryuk itself is not used to gaining access. However, Trickbot or Emotet gains access to the machine, and only later is Ryuk itself deployed. All three families run malicious code, but Ryuk is more specific in the way it goes about running code and only uses three techniques for it, in contrast to Conti, which uses ten different ones. In general, Ryuk usually uses fewer techniques to achieve a certain goal than other ransomware families. This could be due to its very targeted and hidden nature, so the malware authors specifically design the ransomware before every deployment. WannaCry also uses fewer techniques

than Conti to achieve its goals, but this might also be explained by the very different lifetimes of the different families. Conti was actively deployed and updated over two years. In contrast, WannaCry basically only had a single deployment before it was shut down with the registration of the kill switch, and hence, we do not see any evolution. As we already observed in the tags and signatures, WannaCry does more data collection than the other families. Reconnaissance is, however, only done by Conti, and when we analysed this more in detail, we also saw that only 1 or 2 samples in the whole dataset actually use this tactic. So reconnaissance is not standard in ransomware, which also makes sense as ransomware is usually a one-time deployment malware that encrypts files and steals data, and then there is nothing more to do.

Table 4.6: MITRE tactics and techniques for the Conti, Ryuk and WannaCry ransomware families.

TA0001			TA0002			TA0003			TA0004			TA0005			TA0006			TA0007			TA0008			TA0009			TA0011			TA0040			TA0043		
C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W	C	R	W
2	0	2	10	3	6	9	4	7	10	7	9	28	20	21	1	0	7	20	12	13	4	0	1	3	2	6	7	2	8	6	2	3	1	0	0

Note: C stands for Conti, R for Ryuk and W for WannaCry.
The numbers in the cells are the number of techniques used by said ransomware family for the specific tactic.

4.5. Environment

For the comparison between different versions of the same operating system, we used the ANY.RUN sandbox and used their API to submit the samples. Every sample was submitted to Windows versions 7, 8.1, 10 and 11, and the analysis was set to 90 seconds. After the 90 seconds were over, the analysis was stopped, and the results from the automatic analysis were retrieved. We're specifically looking for differences in efficiency between the different OS versions, so we're looking at how long it takes the ransomware to start encrypting the files and how long the encryption process takes. Furthermore, we also look at the tactics and techniques for each version as well as the time it took to trigger specific signatures. We also chose the same environment for every analysis, which is a clean install with no additional programs or files on the sandbox to keep the results as clean as possible. The exact settings for the sandbox setup were:

- OS versions: Windows 7, 8.1, 10, 11
- Bit version: 64
- Environment type: Clean
- Startfolder: Desktop

When comparing different OS versions, we can start by checking how long it takes the ransomware to encrypt the files and drop the ransom notes. In Figure 4.5 and Table 4.7, we compare the times until the first file is encrypted, until the last file is encrypted, when the first ransom note is created and when the final ransom note is created. From the density plots, it becomes clear quickly that the ransomware takes longer for everything on Windows 8 than on any other version. On Windows 8.1, it takes, on average, around 12 seconds longer before Conti starts encrypting the files. The other 3 Windows versions are much closer together, and we cannot really see any significant differences in them. When looking at the duration of the encryption process, we can see that it takes Conti the least amount of time to encrypt Windows 7 systems on average, with a duration of approximately 50 seconds, and Windows 10 systems take the longest, with an average encryption time of around 70 seconds. For the ransom note, we see a very similar trend, with Windows 8.1 taking the longest for the first ransom note to appear but Windows 10 taking the longest until all the ransom notes have been created in each directory.

Next up, we compare the MITRE techniques that were detected in the samples by the ANY.RUN

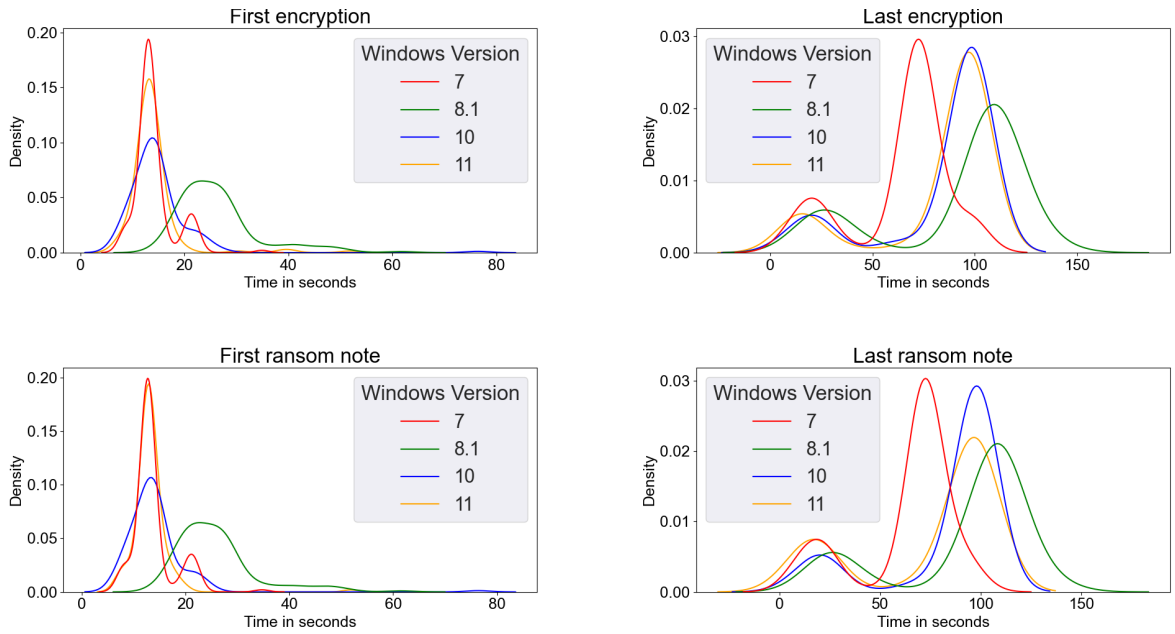


Figure 4.5: Density plots for the first and last encryptions and ransom notes for each Windows version.

sandbox during the analysis. Interestingly, we only found 12 techniques that were used by the samples, whereas the analysis from Virustotal showed that 101 different techniques were detected. We explain this with the different setups of the sandbox as well as with the limited runtime of the sample in each version of Windows. Furthermore, Virustotal also uses different techniques for the analysis of malware and uses a collection of over 70 different scanners, whereas ANY.RUN only uses one of each. Figure 4.6 shows the occurrences of the techniques in the different versions of Windows, and for most techniques, we cannot see a big difference in the numbers. There are, however, a few techniques which only appear in a single version of the OS, or we see a big difference in the number of occurrences. For example, T1021.002 - ”SMB/Windows Admin Shares” is used the same amount of time in Windows 7, 8.1 and 10 but is used much less often in Windows 11. T1490 - ”Inhibit System Recovery” is only used in Windows 7 and 8.1 and cannot be found at all in Windows 10 or 11. T1562.006 - ”Indicator Blocking” can only be seen in Windows 7 and around 75% of all the samples, but not at all in any other version of Windows. Lastly, T1053.005 - ”Scheduled Task” and T1497.003 - ”Time-Based Evasion” can only be found in a single sample, only in Windows 8.1. So overall, we do see some differences in behaviour for some samples in different versions of Windows. This could be due to some changes in how Windows works internally, or Conti has some built-in checks which optimise it for different versions of the OS, as we have previously seen that it also does some reconnaissance at the start of its execution.

We can also observe some differences in the registry between the different versions of Windows. Conti interacts with the registry entries by either reading them, writing to them or deleting them. We collected various statistics about these actions and can then compare Conti’s behaviour across different versions. Conti reads far more registry entries than it modifies or deletes. In Windows 10, Conti interacts with the most registry entries, which are around 4500 entries on average, with a standard deviation of about 9200 entries. Conti makes the fewest interactions with Windows 11, which is surprising, as we thought that Windows 11 has more features and, hence, Conti would profit from reading more registry entries. Still, it seems like the necessary registry entries are less common in Windows 11. Conti also modifies registry

Table 4.7: Times of the first and last encryptions and ransom notes for each Windows version.

	Windows Version	Minimum	Maximum	Average	Median
First encryption	7	8.219	34.877	14.104	13.062
	8.1	15.072	61.524	26.261	26.691
	10	8.153	76.367	14.764	14.016
	11	7.716	51.851	13.939	13.247
Last encryption	7	9.78	98.54	64.169	72.732
	8.1	16.595	145.367	92.005	106.5
	10	8.946	101.866	84.218	98.233
	11	8.319	98.801	82.471	97.184
First ransom note	7	7.624	34.721	13.743	12.757
	8.1	14.666	61.399	25.826	23.953
	10	7.84	76.352	14.061	13.57
	11	7.404	51.836	13.205	12.971
Last ransom note	7	8.819	98.284	63.188	72.609
	8.1	16.517	145.43	92.416	105.952
	10	8.93	102.007	84.535	98.179
	11	8.1	98.446	75.693	96.514

Note: All the times are in seconds.

entries, but the delete and write actions make up only around 1-2% of all the actions. Here, we see a different trend for the delete action, as Conti deletes, on average, more registry entries on Windows 11 than on any other version of Windows. A close second is Windows 10, and Windows 7 and 8.1 make up the smallest amount of deletions. This also shows more similarities between Windows version 7 and 8.1, as well as between Windows 10 and 11. When looking at the write operation, we can see that Conti modifies the most files on Windows 10 and, by far, the least on Windows 11. Windows 7 and 8.1 seem very similar in terms of modifications done to registry values.

Finally, when looking at more of Conti's behaviours over the various versions of the Windows operating system, we can see some more differences. Appendix C shows some graphs which give an overview of this behaviour. Figure C.1 shows some initial reconnaissance Conti does before it starts with any malicious behaviour. This behaviour can be observed in the first 10 seconds for every version of Windows, except for Windows 8.1. As we have already observed before, the start of encryption in Windows 8.1 takes a bit longer than in the other versions. Interstingly, Conti only checks for proxy information on Windows 10 and 11 and not at all on Windows 7 or 8.1. In Figure 4.4, we can see a comparison in file creation behaviour from Conti. The files in the user, root and program directory are created at the same time, and Conti does not seem to start earlier with a specific directory. This makes sense, as Conti multithreads the encryption process and hence can take care of multiple locations at the same time. The first time a ransom note is found also matches with the first encryption detections. As before, in Windows 8.1, it takes a bit longer for this to happen compared to the other versions. In terms of the detection of Conti, Figure C.3 shows the difference in time until Conti is detected from the

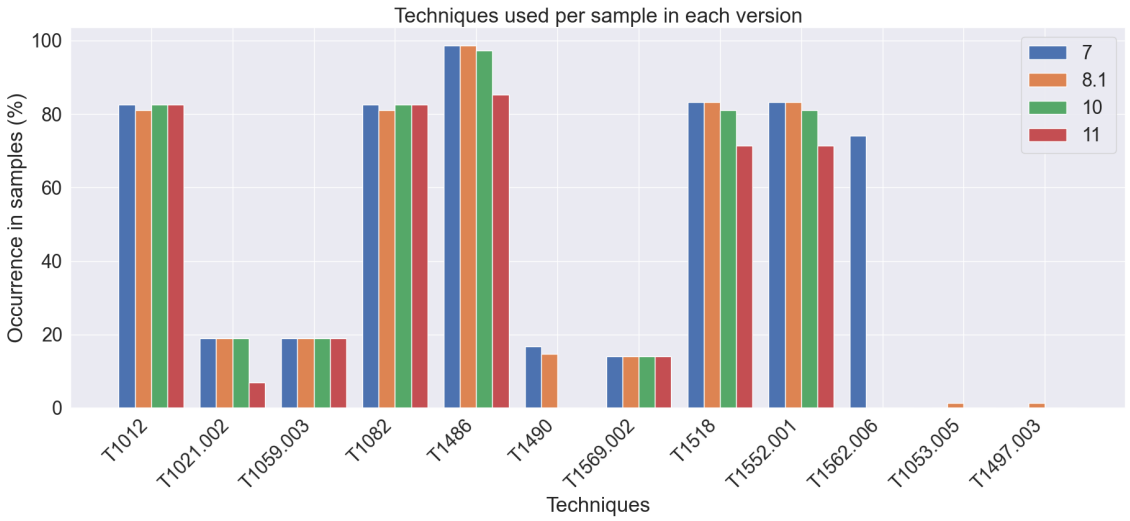


Figure 4.6: The different techniques which were used by the Conti samples on different versions of the Windows operating systems.

moment it is run for the first time. We can see that on Windows 11, the Conti detection signature is not triggered at all. On Windows 7 and 10, there is an initial more minor detection for some samples, whereas the majority is detected around 15 seconds later. The samples on Windows 8.1 were all detected a bit later. The mutex is detected on all versions and dates around the same time on every version of Windows, once again, with the only exception being Windows 8.1, where it is detected a bit later. This also makes sense, as the encryption also only starts later on Windows 8.1. When looking at other, more specific behaviour from Conti in Figure C.4, we can once again see similarities between the versions, with Windows 8.1 being the outlier. The deletion of shadow copies is only being detected on Windows 7 and 8.1, whereas manual inspection shows that it is also happening on Windows 10 and 11. Finally, some miscellaneous behaviour, as can be seen in Figure C.5, shows fewer differences between the versions of Windows. The data exfiltration seems to start taking place around 20 seconds into the start of the executable and is about even for all versions. Trying to cloak folders and files also takes place on all four versions, with Windows 8.1 being around 15 seconds later than the others when changing desktop.ini files.

Table 4.8: Statistics of different operations in the registry across OS versions

Operation	OS Version	Count	Mean	Std	Min	25%	50%	75%	Max
delete	7	143	25.01	16.76	0	0	36	36	48
	8.1	143	24.59	16.75	0	0	36	36	36
	10	143	40.91	29.50	0	0	66	66	66
	11	143	42.84	38.42	0	0	78	78	84
write	7	143	24.99	16.74	0	0	36	36	46
	8.1	143	24.69	16.75	0	0	36	36	37
	10	143	39.75	28.58	0	0	64	64	65
	11	143	6.16	5.34	0	0	11	11	13
read	7	143	2393.78	5099.60	292	295	498	524	16364
	8.1	143	3324.35	7489.99	212	293	513	589	24082
	10	143	4437.49	9205.49	446	651	976	995	29528
	11	143	1726.34	3192.26	147	327	546	563	12285
total	7	143	2443.78	5108.82	292	295	570	596	16436
	8.1	143	3373.63	7499.18	284	293	585	661	24154
	10	143	4518.15	9207.36	446	651	1106	1125	29611
	11	143	1775.34	3176.51	147	327	635	656	12285

5 Discussion

5.1. Summary

In this research, we started by collecting samples of different malware families from two different sources: Malware Bazaar and VXUnderground. Once these samples were acquired, testing for true positives and classification into different variants was the next step. The classification was done with Microsoft Defender Antivirus, as Microsoft has a very convenient variant naming scheme. Once the classification was done, a preliminary analysis showed some irregularities within the data, so a manual inspection of the samples was performed using the ANY.RUN sandbox to verify that all the samples were indeed true positives. It turned out that some of the classifications from Malware Bazaar, VXUnderground and Microsoft Defender Antivirus were not correct and that some samples were wrongly labelled. Some samples belonged to a completely different family, some executables were broken and did not run, and some were not Windows executables. After a second round of data processing, the dataset was verified only to contain true positives, and the analysis could continue. The next step was to extract information about the samples from Virustotal and ANY.RUN and use this data to learn about the differences. The analysis was conducted on a general basis, on a variant basis, and over time. Furthermore, a comparison across different families, as well as a comparison of the samples across different versions of the same OS, was done. This research contributes a lot of processed data about the collected samples and a dataset of true positives for Conti, Ryuk, and WannaCry.

5.2. Results

Conti is generally quite diverse over the samples we have collected. Our central hypothesis, namely that the core stays the same, has proven true. The core of the ransomware stays the same, and the primary behaviour can be observed in all samples. We can see the differences emerge over time or on a variant basis. Still, it is essential to note that there are still slight differences in the core, which can be seen when looking at the MITRE tactics and techniques. Even though most samples try to achieve the same goal, they apply different techniques. This comes from updating and changing the malware over time or finding a workaround for something patched in older versions of the targeted operating system.

On a variant basis, the results were different than we initially expected. We anticipated the results would be more similar if we analysed the same variant. However, the results showed that although some samples were classified as the same variant, they still had very different behaviour. The differences across the variants were expected, and from the way Microsoft

names the malware, we also thought that variants with names closer to one another have similar behaviour. This was mostly true, but not to the extent we expected it. It is, however, hard to draw any conclusions on this, as we're missing quite some information from the Defender, like suffixes and family names, as it is closed-source. All we can say is that the behavioural differences from one variant to another, or even within the same variant, differ from what we expected.

The evolution of the malware over time shows a more expected result than the comparison across the variants. Samples which have a similar creation date also exhibit similar behaviour. We can see the behaviour of the samples evolve and find the same behaviour within samples that were created closer together. Sometimes, we see new behaviour show up in a certain period of time but disappear again, which we can assume is due to a less successful change, which was then discarded. So overall, the comparison over time shows more of the expected results than the comparison between variants and might be the way to go when comparing different generations of the same malware family.

Different ransomware families also show different behaviours, even though we can recognise similarities in the main behaviours of the families. Since we focused our research on Windows ransomware, we can see interactions with the Windows API throughout all the families. However, we see significant differences between Conti and WannaCry, as WannaCry does more enumeration than Ryuk or Conti. Conti is a descendant of the Ryuk ransomware; hence, the hypothesis was that both would have significant similarities, and this hypothesis was also confirmed. Both families show that they are related not only on a structural level with the imports and tags but also on the behavioural level with signatures, tactics and techniques. Ryuk was a very targeted malware and showed these traits in the information we gathered, whereas Conti was more openly and widely used as it is a RaaS. So, although the families show similarities, we also see their differences quite clearly.

Lastly, the OS versions also showed some differences, even if they are only minor. Windows 8.1 stands out from the four versions of Windows that were compared. In Windows 8.1, ransomware takes longer to start and finish the encryption than other versions. We also saw some differences when analysing different malware behaviours on these versions. Sometimes, a particular behaviour does not appear in a specific version. When looking at techniques and registry modifications, we can also see some differences in behaviour. The registry values change at each Windows version, so it makes sense to see differences in the numbers. The techniques showed exciting statistics, as some Windows versions change the malware samples' behaviour or completely stop the samples from showing a specific behaviour.

5.3. Problems encountered

The primary data collection did not pose any problems, even though some families had more samples than others. As we only used open-source databases for this research, the dataset might not be complete if some samples were never published. Additionally, a smaller number of samples for a specific family might also mean that the results might not be complete and that some specific behaviour might not be shown by the samples we collected. If this research is repeated, this can be kept in mind.

The main problem of this research is the large amount of manual inspection needed to verify the data. Even though the data was classified as a certain malware by multiple databases and tools, a manual inspection was needed to confirm that the samples were true positives. This approach is not scalable to a larger dataset, so a more automated approach with a different framework would be needed to extend this research. However, when automating the data

collection and verification, the question remains whether the data only contains true positives or if some classifications were wrong. Wrong classifications could lead to wrong results in the analysis later, leading to the wrong conclusions being drawn. One possibility would be to use more than one automated tool and collect classifications on the same sample before locking in the final result. This way, a less error-prone approach and more signatures would lead to a more likely correct classification.

5.4. Limitations

This research also met some limitations, the first one being, as mentioned before, the number of samples that we found for certain malware families. Ryuk only had seven true positives in our dataset, which could be improved if more datasets were considered for the data collection.

The following limitation is some of the functionality of the ANY.RUN sandbox. Sandboxes are usually complex to set up and bring a lot of options to configure, which is no different for ANY.RUN. However, some functionality was not available in the ANY.RUN sandbox might be available on another sandbox or in the ANY.RUN sandbox in the future. ANY.RUN does not allow the number of threads a malware uses to be seen. In the case of Conti, this is a limitation, as observing how the 32 threads behave would be an interesting insight that could also give some more information about the behaviour compared to non-threaded ransomware.

During the research, we also came up with the idea of using different versions of Defender to see when the malware started to be detected and how different signatures affected the detection rate of said family over time. The ANY.RUN sandbox has the Defender deactivated by default, and only with specific administrative controls can it be reactivated. As we did not have these controls, nor was there any possibility of changing the version of Defender, we could not do this part of our research.

Finally, another limitation was the closed-source nature of Microsoft Defender Antivirus and the limited number of documentation available. As some information, especially about suffixes and the actual malware family names, brought up some questions, we had to make some assumptions during this research. If this information were more openly available, the study could be complemented by this information.

6

Conclusion

Analysing different versions of the same malware provides invaluable insights into its evolution, functionality, and strategies malware authors employ. By comparing successive iterations, we can identify modifications in code, new features, and changes in behaviour, revealing how malware adapts to bypass security measures. This analysis helps understand the core components and behaviour patterns. Additionally, tracking changes across versions can highlight trends in malware development, enabling the prediction of future threats and the creation of more robust defence mechanisms. Ultimately, this process deepens the knowledge of malware architecture and improves the efficacy of detection and mitigation strategies.

The question we asked at the beginning of this research was *"How do different generations of the malware families succeed in bypassing security measures?"* and this was split into three different sub-questions. After collecting and analysing samples from other families across different versions of the same operating system, we can now answer these sub-questions.

To answer the question *"Are there specific trends in evasion tactics that persist across multiple generations or families?"*, we can clearly say yes. We see a trend in changes implemented in the same family, which persists across multiple generations. However, we also see the exact opposite; sometimes, a change appears in a sample, but a few months later, the same is not present anymore. This shows some precise efficiency measures of the malware, where the authors constantly try to update and refine their programs and are testing them in the wild. If a sample proves less successful, it is discarded, and the more successful changes are kept. The same can also be seen when comparing different families. The main similarities were the interaction between the malware and the operating system. This behaviour persisted across all the families and was even implemented mainly in the same way. The main differences can be seen when looking at specifics of the malware, as each family has different smaller goals. For example, Conti actively exfiltrates the data due to its double extortion tactic. In contrast, other malware only encrypts the malware and only communicates to the C&C server for encryption or decryption.

The second question was *"Is there a difference in the behaviour of malware when it is executed on a different version of the operating system? Does this impact the efficiency of the malware?"*. This answer cannot be given in general, as we have to split the answer into multiple parts. The execution time of the malware differs from one version of the operating system to another, but this may be due to how the OS is built and not how the malware operates. We can see a clear time difference in the speed of the malware on Windows 8.1, but every single

version of Windows was encrypted by ransomware. When looking at other behaviours or actions from the ransomware, we can also observe that certain behaviours can only be found in certain versions of the OS and are completely absent in others. Sometimes, the behaviour is present in every version of the OS, but it cannot be detected in certain versions, such as the deletion of shadow copies. Conti deletes the shadow copies in every version of Windows, but it was only detected by the sandbox in Windows 7 and 8.1.

Lastly, the question *"What is the most common behaviour detected across different malware families and their generations, and are different techniques employed to achieve the same goals?"* also has a clear answer. The most common behaviour detected is the interaction with the operating system. This behaviour in itself is not malicious, but a combination of other actions or the way in which this interaction is used is malicious. When a regular user interacts with their operating system, they use the same functions but with a different goal. If there could be a way which, instead of checking every action individually, would consider the actions as a chain, then the detection could be more precise and false positives could be reduced. This would, however, also mean that simply breaking a small part of the chain would lead to a bypass of the detection again, so there is no simple solution to this. When looking at the techniques, we can clearly say that different malware employs different techniques to reach the same goals. We see the same tactics in every analysed malware, but the techniques the malware authors use to achieve their goals differ from one family to another, even over time, within the same family.

So, to answer the main question, we can say that malware authors employ various strategies to bypass security measures and stay ahead of the defences. Malware authors employ and test out different techniques and constantly check which ones are the most successful. The ones that are not successful are discarded, and the best ones are kept. Furthermore, even when a specific behaviour is detected in a sample, the malware authors do not necessarily change the behaviour of the sample; they only use different libraries or functions to do the same but evade the signature-based detection. If the defensive measures cannot predict the changes the cybercriminals implement, they will always be one step ahead. By understanding how evolution works and how different generations differ, we can start predicting the malware and maybe win the endless cat-and-mouse game once and for all.

6.1. Future Work

As mentioned in the research's limitations, some things had to be assumed or taken away from the research. This research focused only on Windows ransomware, but even though Windows is the most prominent operating system, it is not the only system for which malware is made. One step that could be taken in the future would be to look into other operating systems, such as Linux distributions or Apple's MacOS. Additionally, the research could even be extended to malware which targets mobile phones to see if there are any differences.

As malware evolves, we can also see which specific signatures and defensive measures significantly impact the successful detection and blocking of threats. By testing different versions of Microsoft Defender Antivirus or other antivirus scanners, we could develop a timeline and analyse in more detail which changes to our defence have the best impact on security. For this path, it would be crucial to have as much information about the scanners as possible, as our research already showed that a black box implementation with limited documentation leads to assumptions that must be made.

Finally, this research focused solely on ransomware, but many different malware types exist. By not only looking at ransomware but extending the research to various types, we could learn

something more about other threats in the cybersecurity realm. Perhaps this could even lead to a hybrid security approach, encompassing multiple threats into a single security solution.

References

[1] Amir Afianian et al. “Malware dynamic analysis evasion techniques: A survey”. In: *ACM Computing Surveys (CSUR)* 52.6 (2019), pp. 1–28.

[2] *ANY.RUN - Interactive Online Malware Sandbox*. URL: https://any.run (visited on 05/30/2024).

[3] Abdullahi Arabo et al. “Detecting ransomware using process behaviour analysis”. In: *Procedia Computer Science* 168 (2020), pp. 289–296.

[4] Moses Ashawa, Sarah Morris, et al. “Analysis of mobile malware: a systematic review of evolution and infection strategies”. In: (2021).

[5] Ömer Aslan Aslan and Refik Samet. “A comprehensive review on malware detection approaches”. In: *IEEE access* 8 (2020), pp. 6249–6271.

[6] Brian Baskin. *TAU Threat Discovery Conti Ransomware - VMware*. July 2020. URL: https://blogs.vmware.com/security/2020/07/tau-threat-discovery-conti-ransomware.html (visited on 05/30/2024).

[7] *Behaviour MITRE Trees - VirusTotal*. 2022. URL: https://docs.virustotal.com/reference/get-a-summary-of-all-mitre-attck-techniques-observed-in-a-file (visited on 05/30/2024).

[8] Marcus Botacin et al. “Challenges and pitfalls in malware research”. In: *Computers & Security* 106 (2021), p. 102287.

[9] Marcus Felipe Botacin, Paulo Lício de Geus, and André Ricardo Abed Grégio. “The other guys: automated analysis of marginalized malware”. In: *Journal of Computer Virology and Hacking Techniques* 14 (2018), pp. 87–98.

[10] Salvatore Bova. “An evaluation of anti-evasion techniques implemented in malware analysis sandboxes and debuggers”. In: (2020).

[11] Karl Bridge. *PE format - Win32 apps*. Feb. 2024. URL: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (visited on 05/30/2024).

[12] Frank Broy. *Is there a full list of suffixes used in the malware naming for Microsoft Defender Antivirus?* Feb. 2024. URL: https://learn.microsoft.com/en-us/answers/questions/1570876/is-there-a-full-list-of-suffixes-used-in-the-malwa (visited on 05/30/2024).

[13] Frank Broy. *What is the difference between ContiCrypt, Conti and Conti in the naming of Microsoft Defender Antivirus?* Feb. 2024. URL: https://learn.microsoft.com/en-us/answers/questions/1571261/what-is-the-difference-between-conticrypt-conti-an (visited on 05/30/2024).

[14] Emanuele Cozzi et al. “Understanding Linux malware”. In: *2018 IEEE symposium on security and privacy (SP)*. IEEE. 2018, pp. 161–175.

[15] Chuong Dong. *Conti Ransomware Reverse Engineered*. Dec. 2020. URL: https://chuongdong.com/reverse%20engineering/2020/12/15/ContiRansomware/ (visited on 05/30/2024).

[16] Osama Ellahi, Munam Ali Shah, and Muhammad Usman Rana. “The ingenuity of malware substitution: Bypassing next-generation Antivirus”. In: *2021 26th International Conference on Automation and Computing (ICAC)*. IEEE. 2021, pp. 1–5.

[17] David Emm. “Focus on trojans–holding data to ransom”. In: *Network Security* 2006.6 (2006), pp. 4–7.

[18] Dexter Eugenio. *A Targeted Campaign Break-Down - Ryuk Ransomware*. Jan. 2023. URL: https://research.checkpoint.com/2018/ryuk-ransomware-targeted-campaign-break/ (visited on 05/30/2024).

[19] *Files - VirusTotal*. Aug. 2023. URL: https://docs.virustotal.com/reference/files (visited on 05/30/2024).

[20] *Files Behaviour - VirusTotal*. July 2023. URL: https://docs.virustotal.com/reference/file-behaviour-summary (visited on 05/30/2024).

[21] *Ghidra*. URL: https://ghidra-sre.org/ (visited on 05/30/2024).

[22] *GitHub repository - Thesis scripts and data*. URL: https://github.com/FBroy/thesis.

[23] Kyoung Soo Han et al. “Malware analysis using visualized images and entropy graphs”. In: *International Journal of Information Security* 14 (2015), pp. 1–14.

[24] *Hex Rays - State-of-the-art binary code analysis solutions*. URL: https://hex-rays.com/ida-pro/ (visited on 05/30/2024).

[25] *How Microsoft names Malware*. Apr. 2024. URL: https://learn.microsoft.com/en-gb/defender-xdr/malware-naming (visited on 05/30/2024).

[26] Keith Jarvis. *CryptoLocker ransomware threat analysis - Secureworks*. URL: https://www.secureworks.com/research/cryptolocker-ransomware (visited on 05/30/2024).

[27] Danial Javaheri, Pooia Lalbakhsh, and Mehdi Hosseinzadeh. “A novel method for detecting future generations of targeted and metamorphic malware based on genetic algorithm”. In: *IEEE access* 9 (2021), pp. 69951–69970.

[28] Samuel Kim. *PE Header Analysis for Malware Detection*. 2018.

[29] Ya-shu Liu et al. “A new learning approach to malware classification using discriminative feature extraction”. In: *IEEE Access* 7 (2019), pp. 13015–13023.

[30] *MalwareBazaar | Malware sample exchange*. URL: https://bazaar.abuse.ch (visited on 05/30/2024).

[31] Ori Or-Meir et al. “Dynamic malware analysis in the modern era—A state of the art survey”. In: *ACM Computing Surveys (CSUR)* 52.5 (2019), pp. 1–48.

[32] Per Håkon Meland, Yara Fareed Fahmy Bayoumy, and Guttorm Sindre. “The Ransomware-as-a-Service economy within the darknet”. In: *Computers & Security* 92 (2020), p. 101762.

[33] *MITRE ATT&CK®*. URL: https://attack.mitre.org/ (visited on 05/30/2024).

[34] Andreas Moser, Christopher Kruegel, and Engin Kirda. “Limits of static analysis for malware detection”. In: *Twenty-third annual computer security applications conference (ACSAC 2007)*. IEEE. 2007, pp. 421–430.

[35] Pratikkumar Prajapati and Mark Stamp. “An empirical analysis of image-based learning techniques for malware classification”. In: *Malware analysis using artificial intelligence and deep learning* (2021), pp. 411–435.

[36] Matthew Ryan. *Ransomware Revolution: the rise of a prodigious cyber threat*. Springer, 2021.

[37] Sanjay K Sahay, Ashu Sharma, and Hemant Rathore. “Evolution of malware and its detection techniques”. In: *Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2018*. Springer. 2020, pp. 139–150.

[38] John F Shoch and Jon A Hupp. “The “worm” programs—early experience with a distributed computation”. In: *Communications of the ACM* 25.3 (1982), pp. 172–180.

[39] Fabian M Teichmann and Sonia R Boticiu. “The most impactful ransomware attacks in 2023 and their business implications”. In: *International Cybersecurity Law Review* (2024), pp. 1–11.

[40] *VXUnderground*. URL: `https://vx-underground.org/#E:/root` (visited on 05/30/2024).

[41] Anne E Webster. “University of Delaware and the Pakistani computer virus”. In: *Computers & Security* 8.2 (1989), pp. 103–105.

A

List of extracted fields from Virustotal

- creation_date
- detectiteasy
- first_seen_itw_date
- first_submission_date
- last_analysis_results
- last_analysis_stats
- hashes (md5, sha1, sha256)
- magic
- names
- packers
- pe_info
- popular_threat_classification
- sandbox_verdicts
- sigma_analysis_results
- sigma_analysis_stats
- size
- tags
- trid
- type_description
- type_extension
- type_tags
- attack_techniques
- calls_highlighted
- command_executions
- crypto_algorithms_observed
- dns_lookups
- files (files_deleted, files_dropped, files_opened, files_written)
- files_attributes_changed
- ids_alerts
- ip_traffic
- hosts_file
- mitre_attack_techniques
- modules_loaded
- mutexes (mutexes_created, mutexes_opened)
- processes (processes_created, processes_injected, processes_killed, processes_terminated, processes_tree)
- registry_keys (registry_keys_deleted, registry_keys_opened, registry_keys_set)
- signature_matches
- tags
- verdicts
- services (services_bound, services_created, services_deleted, services_opened, services_stopped, services_started)
- signals_observed
- text_decoded
- text_highlighted
- windows_hidden
- windows_searched
- sandbox_name

B

DLLs found in the Conti samples

Table B.1: Dynamically Linked Libraries (DLLs) and the amount of times they were used by the collected Conti samples.

Library Name	Occurrences	Description
kernel32.dll	143	Handles memory management, input/output operations, and various system services essential for the operation of Windows-based applications.
user32.dll	117	Manages user interface components, including the creation and management of windows, input processing, and interaction with the graphical user interface.
shlwapi.dll	98	Provides utility functions for handling file paths, URLs, and registry operations, aiding in various tasks related to file and system operations.
ws2_32.dll	33	Provides functions for network communication, specifically implementing the Windows Sockets API (Winsock) for handling TCP/IP networking operations.
shell32.dll	22	Provides functions for the Windows Shell, including the management of files, folders, and user interface elements like the Start Menu and Taskbar.
ole32.dll	11	Provides support for Object Linking and Embedding (OLE) technology, facilitating communication and interaction between software components through mechanisms like COM and ActiveX.
advapi32.dll	10	Provides advanced services related to security, including authentication, access control, auditing, and cryptographic functions essential for managing system security policies and user privileges.
oleauth32.dll	6	Provides support for OLE Automation, enabling software components to communicate and interact with each other through a standardised set of interfaces and protocols.
comdlg32.dll	2	Provides common dialogue box functionality for various user interactions, such as opening and saving files, choosing printers, and selecting colours.
gdi32.dll	2	Provides functions related to graphics device interface (GDI), including rendering graphical elements, managing fonts, and handling graphical output on display devices.

Table B.2: Dynamically Linked Libraries (DLLs) and the amount of times they were used by the collected Conti samples. (part 2)

Library Name	Occurrences	Description
winspool.drv	2	Provides functions for interacting with printers and managing print jobs, including sending print commands, configuring printer settings, and monitoring printer status.
oledlg.dll	2	Provides common dialogue functionality related to Object Linking and Embedding (OLE), facilitating interactions such as embedding and linking objects within applications through standardised dialogue interfaces.
imagehlp.dll	1	Provides functions for manipulating and examining Portable Executable (PE) files, including tasks such as extracting information about modules, debugging, and manipulating executable images.
msimg32.dll	1	Providing functions related to image rendering and manipulation, including support for advanced image processing operations such as alpha blending and gradient filling.
comctl32.dll	1	Provides common controls and user interface elements for Windows applications, including functionalities such as buttons, list views, tree views, progress bars, and tab controls, enhancing the graphical user interface (GUI) of applications.
winmm.dll	1	Provides functions related to multimedia support, including audio playback, recording, and MIDI (Musical Instrument Digital Interface) input/output operations, enabling applications to interact with multimedia devices and resources.

C

Detected behaviour from Conti samples in ANY.RUN sandbox

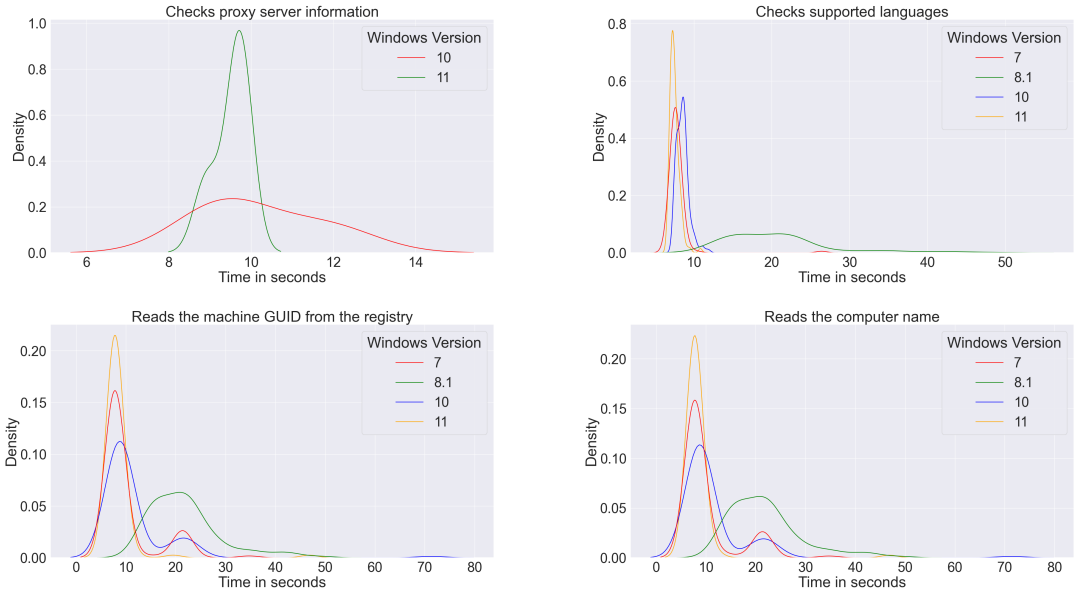


Figure C.1: Conti reconnaissance on different Windows versions.

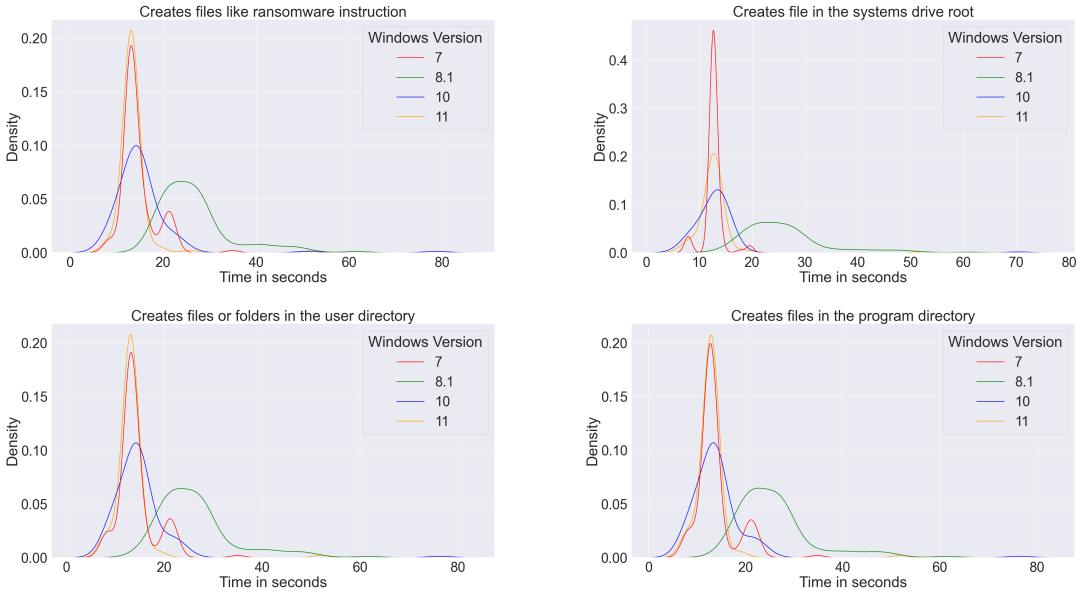


Figure C.2: Conti file creation on different Windows versions.

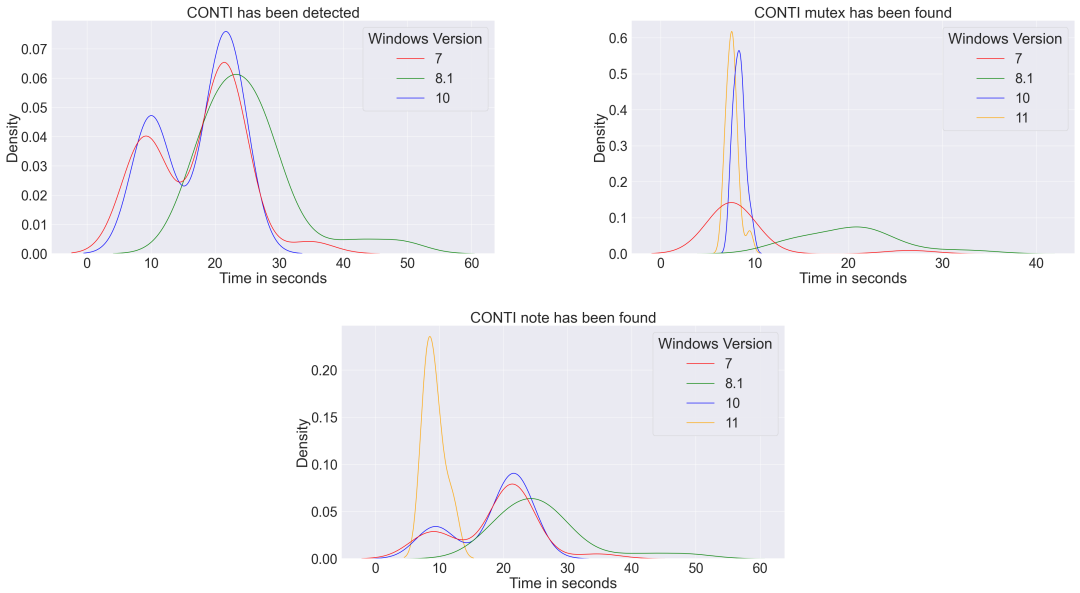


Figure C.3: Conti detection on different Windows versions.

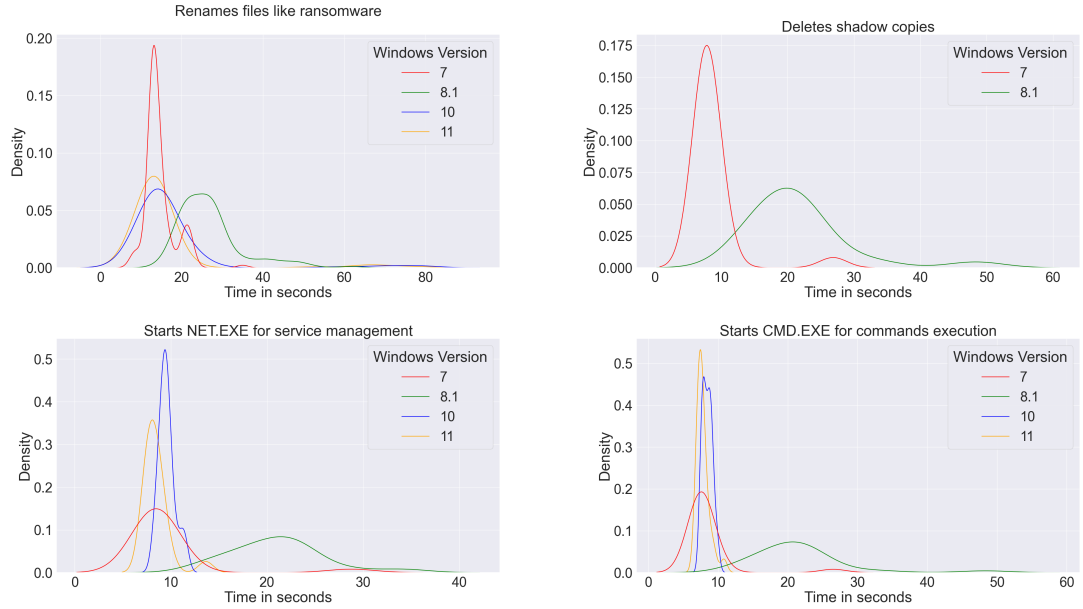


Figure C.4: Conti actions on different Windows versions.

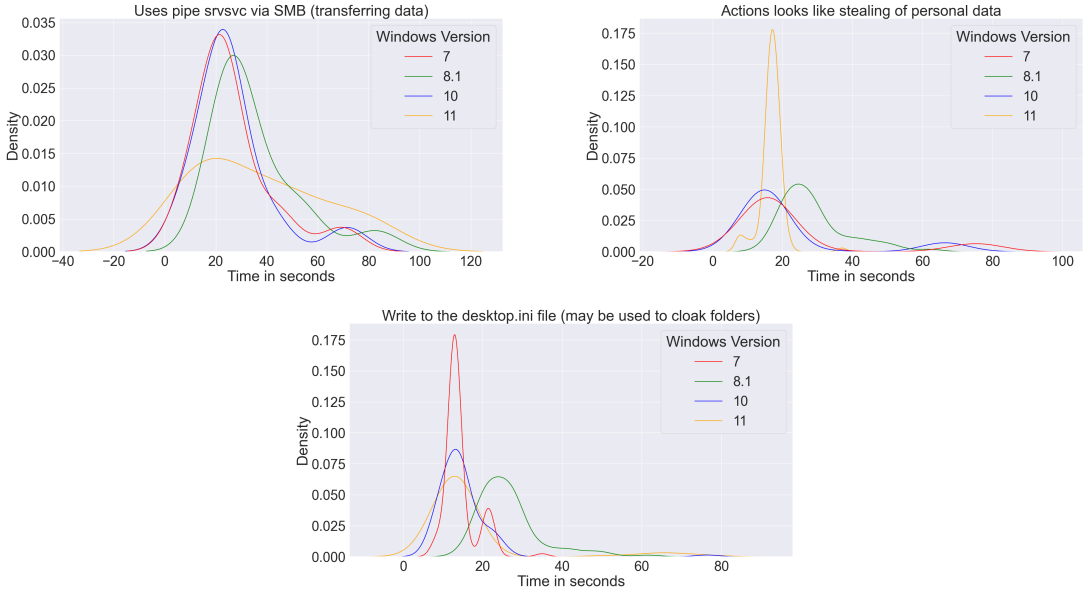


Figure C.5: Conti miscellaneous on different Windows versions.

D

DLL comparison between Conti, Ryuk and WannaCry

Table D.1: DLLs used by the Conti, Ryuk and WannaCry ransomware families.

DLL	Conti	Ryuk	WannaCry
advapi32.dll	x	x	x
api-ms-win-crt-filesystem-l1-1-0.dll		x	
api-ms-win-crt-heap-l1-1-0.dll		x	
api-ms-win-crt-locale-l1-1-0.dll		x	
api-ms-win-crt-math-l1-1-0.dll		x	
api-ms-win-crt-runtime-l1-1-0.dll		x	
api-ms-win-crt-stdio-l1-1-0.dll		x	
avicap32.dll		x	
avifil32.dll		x	
comctl32.dll	x	x	
comdlg32.dll	x		
crypt32.dll		x	
dciman32.dll		x	
gdi32.dll	x	x	
glu32.dll		x	
imagehlp.dll	x		
iphlpapi.dll		x	x
kernel32.dll	x	x	x
msimg32.dll	x	x	
msvcp140.dll		x	
msvcp60.dll			x
msvcrt.dll			x
netapi32.dll		x	

Table D.2: DLLs used by the Conti, Ryuk and WannaCry ransomware families. (part 2)

DLL	Conti	Ryuk	WannaCry
odbc32.dll		x	
ole32.dll	x	x	
oleaut32.dll	x	x	
oledlg.dll	x		
opengl32.dll		x	
secur32.dll		x	
shell32.dll	x	x	
shlwapi.dll	x	x	
user32.dll	x	x	x
usp10.dll		x	
vcruntime140.dll		x	
vcruntime140_1.dll		x	
wininet.dll		x	x
winmm.dll	x		
winspool.drv	x		
ws2_32.dll	x	x	x