



Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Discovery & Mitigation of Cryptojacking in Cloud Systems using Honeypots

EDVIN BAGGMAN

Discovery & Mitigation of Cryptojacking in Cloud Systems using Honeypots

EDVIN BAGGMAN

Degree Programme in Computer Science and Engineering

Date: June 13, 2024

Supervisors: Jose Berengueres, Linus Ringvall

Examiner: Johan Håstad

School of Electrical Engineering and Computer Science

Host company: Subset AB

Swedish title: Upptäckt & Begränsning av Cryptojacking i Molnsystem med hjälp av Honeypots

Abstract

Blockchain-based cryptocurrencies, some of which require resource-intensive crypto mining through Proof-of-Work algorithms, have led to the emergence of cryptojacking, where unauthorized computing resources are exploited to mine cryptocurrency. Concurrently, a significant shift towards cloud services with virtually infinite resources has presented new opportunities for attackers: host-based cryptojacking on cloud systems. This poses significant risks, including increased operational costs and degraded user performance.

Due to the limited research on host-based cryptojacking, this thesis aims to investigate the prevalence and patterns of cryptojacking attacks on cloud systems. By deploying honeypots across major cloud service providers, data is collected and analyzed to understand the threat landscapes across cloud service providers and the patterns of cryptojacking attacks.

The results reveal notable differences in the threat landscape among cloud service providers, with variations in attack frequency and patterns. The analysis of the 26 caught cryptojacking malware samples suggests they belong to two distinct cryptojacking families. Following the analysis, YARA rules, a pattern-matching tool for malware detection, were developed to identify these cryptojacking families. These rules displayed great performance when applied to a constructed dataset of cryptojacking malware samples.

The conclusion drawn from this study is that two cryptojacking families had a notable attack prevalence on cloud systems during the period of the study. Furthermore, the study provides a concept of how honeypot data can be leveraged to help mitigate cryptojacking attacks through the development of YARA rules. These findings highlight the need for adequate detection and mitigation strategies to protect cloud systems from cryptojacking attacks.

Keywords

Honeypot, Security, Network, Cloud computing, Crypto mining, Cryptojacking, YARA

Sammanfattning

Blockkedjebaserade kryptovalutor, varav vissa kräver resursintensiv mining ("valutagrävning") genom Proof-of-Work algoritmer, har lett till framväxten av cryptojacking, där obehöriga datorresurser utnyttjas för mining. Samtidigt har en signifikant övergång till molntjänster, som praktiskt taget har obegränsade resurser, skapat nya möjligheter för angripare: värdbaserad cryptojacking på molnsystem. Detta innebär betydande risker, inklusive ökade driftkostnader och försämrad användarprestanda.

På grund av den begränsade forskningen på värdbaserad cryptojacking syftar denna studie till att undersöka förekomsten och mönstren av cryptojacking-attacker på molnsystem. Genom att sätta upp honeypots ("honungsfällor") på större molntjänstleverantörer samlas data in och analyseras för att förstå hotlandskapen över molntjänstleverantörer och mönstren av cryptojacking-attacker.

Resultaten avslöjar betydande skillnader i hotlandskapet bland molntjänstleverantörer, med variationer i attackfrekvens och mönster. Analysen av de 26 fångade cryptojacking filerna tyder på att de tillhör två distinkta cryptojacking familjer. Efter analysen utvecklades YARA regler, ett mönstermatchningsverktyg för detektering av skadlig programvara, för att identifiera dessa cryptojacking familjer. Reglerna visade god prestanda när de tillämpades på ett konstruerat dataset av cryptojacking filer.

Slutsatsen som dras från denna studie är att två cryptojacking familjer hade en anmärkningsvärd attackförekomst på molnsystem under studieperioden. Vidare ger studien ett koncept om hur data från honeypots kan utnyttjas för att hjälpa till att begränsa cryptojacking-attacker genom utveckling av YARA regler. Dessa fynd understryker behovet av lämpliga detekterings- och begränsningsstrategier för att skydda molnsystem från cryptojacking-attacker.

Nyckelord

Honeypot, Säkerhet, Nätverk, Molntjänster, Kryptovalutagrävning, Cryptojacking, YARA

Acknowledgments

I would like to thank my supervisor, Linus Ringvall, at the host company Subset AB for his ideas and expertise, ensuring the thesis was kept on the right path. I would also like to thank my supervisor at KTH, Jose Berengueres, for providing valuable assistance in completing the project. Lastly, I would like to thank both Philip Joelsson and Amadéus Eskengren for being by my side during the entire education at KTH, helping me every step of the way.

Stockholm, June 2024

Edvin Baggman

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem	2
1.2.1	Research Question	2
1.3	Purpose	3
1.4	Goals	3
1.5	Research Methodology	4
1.6	Delimitations	4
1.7	Structure of the thesis	5
2	Background	7
2.1	Cloud Computing	7
2.1.1	Security in Cloud Computing	8
2.2	Honeypots	9
2.2.1	Types of Honeypots	10
2.2.2	Legal Issues	10
2.2.3	The Cowrie Honeypot	11
2.2.3.1	Cowrie Deceptiveness	11
2.3	Emergence of Cryptojacking	12
2.3.1	Cryptocurrencies and Blockchain	12
2.3.2	Crypto Mining	13
2.3.3	Cryptojacking	14
2.3.3.1	Types of Cryptojacking	14
2.4	Malware Detection	16
2.4.1	Malware Packing	17
2.4.2	Detecting Cryptojacking	18
2.4.3	Yet Another Recursive Acronym	19
2.4.3.1	Evaluating YARA Rules	20
2.5	Related work area	20

2.5.1	Kelly et al.	20
2.5.2	Jayasinghe and Poravi	21
2.5.3	Zimba et al.	22
2.5.4	Baser et al.	23
3	Method	25
3.1	Research Process	25
3.2	Data Collection	25
3.2.1	Cowrie Deployment	27
3.2.1.1	Virtual Machine Configuration	27
3.2.1.2	Cowrie Configurations	27
3.2.1.3	Cowrie Honeypot Obscurer	28
3.2.2	Validity and Reliability of Data	29
3.3	Methods for Analysing Honeypot Logs	29
3.3.1	Attack Source	29
3.3.2	Attack Frequency	30
3.3.3	Attack Evolution	30
3.3.4	Propagation of Attacks	31
3.3.5	Attack Patterns	31
3.4	Methods for Malware Analysis	31
3.4.1	Fingerprinting	32
3.4.2	Packer Detection	33
3.4.3	String Extraction	33
3.4.4	File Parsing	33
3.4.5	Disassembly	33
3.5	Mitigation of Cryptojacking Attacks	34
3.5.1	Development of YARA Rules	34
3.5.2	Evaluation of YARA Rules	34
3.6	Ethical Considerations	34
4	Results and Analysis	37
4.1	Analysis of Honeypot Logs	37
4.1.1	Attack Source	37
4.1.2	Attack Frequency	38
4.1.3	Attack Evolution	41
4.1.4	Propagation of Attacks	41
4.1.5	Attack Patterns	43
4.2	Analysis of Malware	44
4.2.1	Fingerprinting	44

4.2.2	Packer Detection	45
4.2.3	String Extraction	46
4.2.4	File Parsing	47
4.2.5	Disassembly	47
4.2.6	Malware Analysis Summary	47
4.2.6.1	Redtail	48
4.2.6.2	Panchan	48
4.3	Mitigation of Cryptojacking Attacks	49
4.3.1	Development of YARA Rules	49
4.3.2	Evaluation of YARA Rules	51
4.4	Validity and Reliability Analysis	52
5	Discussion	55
5.1	Overall Results	55
5.2	Attacks on Cloud Service Providers	55
5.3	Cryptojacking on Cloud Systems	56
5.4	Mitigation of Cryptojacking Attacks	58
6	Conclusions and Future work	61
6.1	Conclusions	61
6.2	Limitations	62
6.3	Future work	62
6.4	Reflections	63
	References	65
A	Supporting materials	73
B	Control-flow graph of entry point for redtail-files	75

List of Figures

2.1	Example of placement of honeypot in a network	10
2.2	Lifecycle of browser-based cryptojacking	15
2.3	Lifecycle of host-based cryptojacking	16
3.1	Research Process	26
3.2	Static malware analysis process flow	32
4.1	Origin of unique IP addresses attacking the cloud service providers	38
4.2	Venn diagram of unique IP addresses observed attacking the cloud service providers	39
4.3	Propagation Graph over the cloud service providers. The graph shows the likelihood of an observed IP address later being observed at the target honeypot. In the first graph, the whole 4-week period is used. In the second graph, the periods are split.	42
4.4	Symbolic aggregate approximation (SAX) on Google Cloud Platform (GCP), Amazon Web Services (AWS) and Microsoft Azure (Azure). The figure describes the number of unique sessions in each 6-hour chunk over the 4-week period for each cloud service provider. The breakpoints for SAX are also shown.	43
4.5	Origin countries of cryptojacking attacks	45
B.1	Control-flow graph of entry point for redtail-files for ARM	75
B.2	Control-flow graph of entry point for redtail-files for AMD64	75
B.3	Control-flow graph of entry point for redtail-files for i386	76
B.4	Control-flow graph of entry point for redtail-files for AARCH64	76

List of Tables

3.1	Configurations used on virtual machines	27
4.1	Total number of sessions, commands executed, unique IP addresses and downloaded files on the platforms	40
4.2	Number of sessions, commands executed, unique IP addresses and downloaded files per day on the platforms	40
4.3	Number of downloaded files per session, commands executed per session and sessions per IP address on the different platforms	40
4.4	Breakpoints for bins used in SAX	43
4.5	Results from SAX for the cloud service providers	44
4.6	Summary of found cryptojacking malware divided on the cloud service providers	44
4.7	Summary of found cryptojacking malware	45
4.8	Results of unpacking the Ultimate Packer for eXecutables (UPX)-packed cryptojacking malware	46
4.9	Detection results of YARA rules	52
4.10	Evaluation metrics of YARA rules	52

Listings

2.1	Example of a YARA-rule	19
3.1	Configurations used on the Cowrie honeypot (cowrie.cfg)	28
4.1	Entry in cowrie.json describing the occurred incident	37
4.2	YARA rule to detect Redtail cryptojacking malware	50
4.3	YARA rule to detect Panchan cryptojacking malware	51

List of acronyms and abbreviations

AWS	Amazon Web Services
Azure	Microsoft Azure
CPU	Central processing unit
DIE	Detect It Easy
ELF	Executable and Linkable Format
GCP	Google Cloud Platform
GPU	Graphics processing unit
IaaS	Infrastructure as a Service
IoC	Indicator of Compromise
OS	operating system
PoS	Proof-of-Stake
PoW	Proof-of-Work
SAX	Symbolic aggregate approximation
UPX	Ultimate Packer for eXecutables
VM	Virtual machine
VPN	Virtual private network
YARA	Yet Another Recursive Acronym

Chapter 1

Introduction

This chapter constitutes an introduction, setting the context for the thesis. It includes a quick background and the research questions.

1.1 Background

Cloud computing has revolutionized how organizations and individuals store, process, and manage data, making it an essential part of modern digital infrastructure [1]. However, the widespread adoption of cloud services has also introduced new vulnerabilities and expanded the attack surface for cybercriminals [2]. Among these cyber threats, cryptojacking has emerged as a significant concern [3].

Cryptojacking involves the unauthorized use of computing resources to mine cryptocurrency, often resulting in performance degradation, increased operational costs, and compromised security. Like many forms of cybercrime, the primary motive is financial profit, but unlike other malware, such as ransomware, it is designed to remain hidden from the victim [3]. It is estimated that cryptojacking affected 9% of corporate networks in 2023 [4], which is a decrease from previous years, 16% in 2022 [5], 19% in 2021 [6], presumably due to Bitcoin rates not rebounding to their 2021 peak. Now that Bitcoin reached a new all-time high in March 2024, we could expect an increase in cryptojacking attacks.

Host-based cryptojacking is malware attackers employ to access the host resources for crypto mining. Compared to browser-based cryptojacking malware, host-based malware does not access the computation power through a web script; instead, it must be installed on the host system. The cloud, with its host-based capabilities, virtually infinite resources, and large attack

surface, makes a good target for host-based cryptojacking [3]. Browser-based cryptojacking has seen some attention in research, but host-based cryptojacking is an understudied subject, with little contribution in recent years [7].

A concern for cloud environments is that cyber threats change rapidly, and attacks are improving daily. Traditional security measures struggle to keep up, making innovative defence mechanisms essential [8]. Honeypots, decoy systems designed to lure and analyze attackers, offer a promising approach to understanding and mitigating cyber threats, including cryptojacking [9]. By simulating vulnerable cloud environments, honeypots can attract attacks, providing valuable insights into attack patterns and the effectiveness of existing security solutions.

This thesis aims to explore the utilization of honeypots as a tool for identifying, analyzing, and mitigating cryptojacking attacks in cloud environments. This research tries to understand the prevalence and patterns of cryptojacking attacks by studying the threat landscapes across different cloud service providers. As digital infrastructure continues to evolve, ensuring its security against emerging threats like cryptojacking is essential for maintaining the integrity, performance, and reliability of cloud services.

1.2 Problem

The problem this thesis addresses is the limited amount of research on how prevalent cryptojacking attacks are on cloud systems. Notably, various studies utilizing honeypots either entirely overlook the issue of cryptojacking or presume that other categories of attacks hold greater significance. While some studies acknowledge the existence of cryptojacking attacks on cloud systems, they often show an inadequate understanding of the attacks. Without knowing the prevalence and patterns of cryptojacking attacks on cloud systems, developing effective mitigation strategies, let alone conducting a simple risk audit, becomes challenging.

1.2.1 Research Question

The research questions of this thesis are:

RQ 1 *What are the differences between the threat landscapes found by honeypots across different cloud service providers?*

RQ 2 *What are the prevalence and patterns of cryptojacking attacks on cloud systems?*

RQ 3 *What is a suitable method for utilizing honeypot data to improve security measures against cryptojacking?*

RQ 2 is the primary research question of this thesis that affects the methodology the most and addresses the original problem. It is placed second to follow the structure of the thesis better. RQ 1 is a secondary research question posed to obtain additional knowledge from the research. RQ 3 is included to investigate if the result from RQ 2 could help in the general question of how to protect against cryptojacking attacks.

1.3 Purpose

The primary purpose of this thesis is to investigate the amount of cryptojacking attacks in the current threat landscape on cloud service providers. This aims to help users of cloud service providers secure their services against unethical attacks and limit the number of cryptojacking attacks they get exposed to. An additional purpose is to provide a basis for similar future studies focusing on other cloud service providers or other types of attacks.

1.4 Goals

The goal of this thesis is divided into the following four sub-goals:

1. Find a suitable state-of-the-art honeypot for evaluating the research questions. This is preferably a honeypot that can handle downloaded payloads.
2. Deploy the honeypot to suitable cloud providers. The honeypots should be deployed with configurations so they won't be recognized as honeypots.
3. Analyze logs and downloaded payloads on the honeypots to see if the threat landscape differs between cloud service providers and how prevalent cryptojacking attacks are on cloud systems.
4. Use the findings to contribute to the security research community in detecting cryptojacking on cloud systems.

1.5 Research Methodology

The Cowrie honeypot [10], a medium to high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker [11], was deployed to three different cloud providers. The cloud providers, **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)** and **Microsoft Azure (Azure)**, were chosen as they are the three most prominent cloud service providers [12]. After 4 weeks, to answer RQ 1, the honeypot logs were analyzed using proven analysis techniques. To answer RQ 2, the downloaded files on the honeypots were analyzed using a static malware analysis process flow. Lastly, the results from the malware analysis are used to contribute to current solutions to detect cryptojacking, answering RQ 3.

1.6 Delimitations

The study has the following delimitations:

- Not inventing or improving current honeypots. This delimitation is primarily practical, as it would be unnecessary to reinvent the wheel.
- Not using different types of honeypots or making a comparison between honeypots. This delimitation is set to focus the research towards understanding the threat landscape of cloud services.
- Not using different configurations on the honeypots or comparing attacks using different honeypot configurations. This delimitation is set due to time constraints.
- Not using honeypots on personal networks or comparing the threat landscape between personal networks and cloud services. This delimitation is set to focus the research towards understanding the threat landscape of cloud services.
- Not setting up honeypots on cloud services outside of Europe or comparing the threat landscape between different regions. This delimitation is set to ensure legal compliance.
- Not focusing on other attacks than cryptojacking attacks. This delimitation is set to focus the research towards understanding the prevalence and patterns of cryptojacking attacks.

1.7 Structure of the thesis

Chapter 2 presents relevant information about cloud computing, honeypots, crypto mining and malware detection. Next, Chapter 3 presents the methodology used to investigate the problem of the thesis, followed by Chapter 4, which presents the investigation results. In Chapter 5, some discussion of the results in connection to the research questions is made. Finally, Chapter 6 concludes the thesis and informs about future work.

Chapter 2

Background

This chapter covers the background information needed to understand the context of this project. Section 2.1 presents Cloud Computing, Section 2.2 presents the concept of Honeypots, Section 2.3 presents Cryptojacking, and Section 2.4 presents Malware detection. Finally, Section 2.5 informs about related works.

2.1 Cloud Computing

Cloud computing, often referred to as **Infrastructure as a Service (IaaS)**, has gained massive attention over the past decades because of continuously increasing demands [2]. A recent trend for companies has been to move IT infrastructures from private data centres to specialized cloud providers [13]. A cloud service provider is defined as a person or organization responsible for making a cloud service available to a cloud consumer [1] [2]. The largest cloud providers are **Azure**, **AWS**, and **GCP**. These companies are known as the "big three" [12]. AWS is the most significant player with the highest market share in the cloud computing markets [14].

There are several advantages to organizations moving toward cloud-based data storage solutions. These include simplified IT infrastructure and management, remote access from effectively anywhere in the world with a stable internet connection and the cost efficiencies that cloud computing can bring [2].

2.1.1 Security in Cloud Computing

Although there are many benefits to cloud computing, it is not without its own risks and challenges. Cloud computing infrastructure suffers from various security issues arising from the existing and new threats [2]. Security is a significant challenge in adopting the cloud [2]. Cloud computing, by abstraction, prevents the customer from having the same level of influence over the computing resource. In this regard, the cloud customer loses administrative power, operational control, and security control over the cloud system [1]. Data breaches and successful attacks can occur due to lacking identity access management systems, failure to use multi-factor authentication, and weak passwords [1]. The potential consequences of such attacks are severe, given that the cloud offers unlimited computing power. Unauthorized initiation of servers can result in substantial financial liabilities.

The emergence of shared resources in cloud computing has led to systems from different organizations being situated closely together, thereby introducing a new potential target for attacks [1]. Malware that targets the cloud is getting more prevalent as attackers understand the potential of the cloud environment, and every cloud consumer must be aware of new rising threats and corresponding vulnerable areas. Handing over personal data to another party is risky, and users need to understand the related risks [15]. Many companies are not acknowledging the perception gap in cloud security and are vastly underestimating today's threats, leaving themselves vulnerable [2].

Cloud computing utilizes many **Virtual machines (VMs)** and different kinds of servers, where **operating systems (OSes)** play an important role in the security of the cloud [2]. Linux is the most widely used operating system in the cloud, and popular Linux distributions include Redhat, CentOS and Debian [16].

A server needs an external IP address to access and communicate with the server remotely. Cloud service providers have publicly available IP ranges for their services, and an attacker can reliably predict cloud IP address allocations [17]. Consequently, you risk being exposed to attacks by having a cloud server. Additionally, since the IP ranges of cloud service providers are public, attackers can focus their attacks on servers from a specific provider.

2.2 Honeypots

Efficient network security largely relies on understanding both current and emerging threats. Attackers are often a step ahead and will always find ways to bypass the infrastructure and create new attack patterns [18]. To protect systems and their users, it is crucial to collect accurate, concise, high-quality information about malicious activities [19]. Firewalls and Intrusion Detection Systems have difficulties in identifying new attack methods adopted by attackers [20], so relying only upon these traditional lines of defence alone does not provide a good enough coverage on detecting new and emerging patterns of attacks [19].

It is necessary to learn and investigate how cyber criminals scan and interact with systems that are accessible publicly. To understand such activity, honeypots are often deployed [21]. The concept of honeypots is not new and was already used implicitly in the 1990s in the field of information protection and network defences [19], and the term “honeypot” has been established for more than two decades [22]. A honeypot itself is a security instrument that collects information on cyber attacks. Honeypots are purposefully designed so that they can be attacked by malicious actors [23]. By deploying security resources with flaws, honeypots proactively induce hackers to attack it to record the attack information and reveal their motives and identities [20]. Importantly, if any connection is established to a honeypot, it is suspicious by nature [22]. Detecting attacks on production systems is challenging because they blend into the vast amount of production activity. Honeypots can simplify the detection process. Since honeypots have no production activity, all connections to the honeypot are suspect [19].

To learn more about the reasons and methods of such attacks, honeypots can be deployed to capture the interaction of attackers [13]. When researchers deploy honeypots, they can learn more about the techniques and procedures applied by attackers, map current threats and malware and apply such information to improve safety tools and rules [24]. Using honeypots is not an uncommon practice. Large corporations widely employ honeypot technology to safeguard their confidential data on servers. As the prominence of cloud environments continues to expand, honeypots are predicted to become an increasingly vital component within an organization’s security infrastructure [25]. Honeypots add little value directly to the prevention of attacks as they do not protect against security breaches, but if the detection of attacks is concerned, honeypots add extensive value [19].

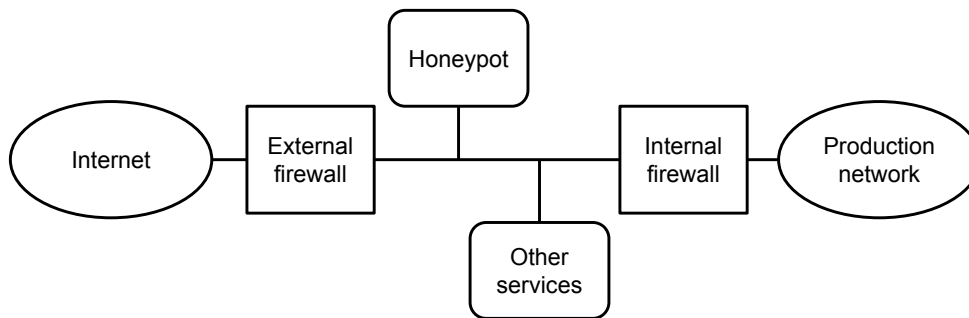


Figure 2.1: Example of placement of honeypot in a network

2.2.1 Types of Honeypots

A widely used classification of honeypots is based upon interaction characteristics [19].

- Low-interaction honeypots
- Medium-interaction honeypots
- High-interaction honeypots

Low-interaction honeypots simulate only a small set of services and provide no access to the operating system to the attacker. As the collection of information is limited, low-interaction honeypots are used mainly for statistic evaluation. Medium-interaction honeypots are slightly more sophisticated. However, they still do not provide any operating system functionality. The simulated services are more elaborated and offer more interaction for the attacker. It produces reasonable replies in the hope of triggering follow-up attacks. High-interaction honeypots are the most sophisticated honeypots. They are the most complex to implement, deploy, and maintain, as they provide the attacker with a real operating system environment that is not restricted. Modern honeypots need a certain degree of interaction to be effective, as they must remain plausible and credible without being detected as a honeypot by their attackers [23].

2.2.2 Legal Issues

To address the questions about what legally is allowed to be collected by honeypots and the legal conditions for the data collection and retention, a study was made, focusing on the European Union regulations [26]. It came to

the conclusion that for research honeypots, *research and prevention of future threats* may be considered a relevant purpose of personal data processing within honeypots. Therefore, they have concluded that administrators of honeypots have a legal ground of legitimate interest in storing and processing personal data, such as IP addresses. Regarding data retention, the data minimisation principle must be considered, and any published data on research honeypots needs to be anonymized.

2.2.3 The Cowrie Honeypot

Cowrie is a medium to high interaction SSH and Telnet honeypot publicly available on GitHub [10], designed to log brute force attacks and the shell interaction performed by the attacker [11]. In medium interaction mode it emulates a UNIX system in Python, in high interaction mode it functions as an SSH and telnet proxy to observe attacker behavior to another system.

In emulated shell mode, the commands are emulated but not executed. The usernames and passwords to access the honeypot are customizable before deployment. After gaining access to the system, the attacker is presented with a fake file system that can add/remove files. Each visitor will get their own personal copy of this file system, which will be deleted when they disconnect. Cowrie saves files downloaded with wget/curl or uploaded with SFTP/SCP for later inspection.

2.2.3.1 Cowrie Deceptiveness

Most honeypots lack the required realism to engage and fool human attackers. This limitation makes them easily discernible as honeypots, hindering their effectiveness [27]. For an expert, it is not hard to distinguish honeypots as bait devices [28]. There needs to be a carefully set balance between the degree of deception and the vulnerabilities present in the honeypot. If too many vulnerabilities are present, the attacker may realize the target is too easy and hence it is a possible trap, if there are too few vulnerabilities, the attacker may not attack the honeypot system at all [23]. One of the key success criteria for a honeypot is that it is indistinguishable from a real system. However, several researchers have shown methods for fingerprinting honeypots over the past years. Honeypot fingerprinting is the process of revealing that a seemingly vulnerable system is, in fact, a honeypot. This significantly decreases the value of a honeypot [29]. Furthermore, honeypot deployments with default configuration settings are easier to detect because of known scripts and tools for identifying them [30].

The current issue with the default Cowrie configuration is that adversaries easily detect it using automated scripts and tools [23]. Problematically, most of the deployments of Cowrie use only the default configurations [30]. The Cowrie documentation also recommends changing the setup, specifically, the default user *Phil* to make the identification of a Cowrie honeypot harder [11]. Fortunately, it is made easy to configure the Cowrie honeypot. Tools available can automate this task, using random values for many of the configurable parameters [31]. Configured Cowrie honeypots provide an improvement in the information collected from attackers, such as an increase in distinct IP connections, the types of commands being executed and files downloaded [30].

2.3 Emergence of Cryptojacking

This section introduces crypto mining. It starts with cryptocurrencies in general and ends with the problem of cryptojacking.

2.3.1 Cryptocurrencies and Blockchain

Cryptocurrency is a peer-to-peer digital exchange system in which cryptography is used to generate and distribute currency units [32]. The first blockchain-based cryptocurrency was Bitcoin, first announced in the famous paper by Satoshi Nakamoto [33]. Today, countless amounts of blockchain and cryptocurrency systems, applications, and technologies are widely available [3].

Blockchains provide a distributed and decentralized environment for transactions of emerging cryptocurrencies, including Bitcoin [34]. A blockchain is a public register of all transactions ever executed. It consists of a distributed, chronological chain of blocks constantly growing as completed blocks are added with a new set of records. A linear path from the first block ever posted to the current block exists because every block includes the hash calculated from the contents of the block itself and the block before it [33]. The consequence of this is that if somebody changes a block in the chain, that change would have to be propagated in the entire chain, which has the implication that individual blocks cannot be altered by malicious actors without it being discovered. Therefore, all the transaction records of cryptocurrencies are considered irreversible and publicly accessible [34].

Most existing blockchains leverage the computationally expensive **Proof-of-Work (PoW)** consensus mechanism. **PoW** assumes that each peer votes

with his computing power by solving **PoW** instances and constructing the appropriate blocks [35]. While it is essential for appending blocks to the blockchain, the process of hash calculation in **PoW** does not serve a fundamental purpose within the algorithm itself, as it doesn't compute anything of particular significance. It is only a required step to make sure transactions are valid. In short, it is a computationally expensive consensus mechanism that allows anonymous entities in decentralized networks to trust one another.

Bitcoin [33] employs a hash-based **PoW**, which entails finding a nonce value, such that when hashed with additional block parameters, the value of the hash has to be smaller than the current target value. When such a nonce is found, the miner creates the block and forwards it to the network layer [35].

Some cryptocurrencies instead use the **Proof-of-Stake (PoS)** consensus mechanism, which was created as an alternative to **PoW** and is less computationally expensive. In **PoS**, the next block writer on the blockchain is selected at random, with higher odds being assigned to nodes with larger stake positions.

2.3.2 Crypto Mining

Cryptocurrencies require distributed verification of transactions without a central authority. In transaction verification that involves the concept of **PoW**, anyone can take part in the process, and this verification process is called cryptocurrency mining [32], also known as crypto mining. Each miner will be rewarded a nominal amount of cryptocurrency if that miner is the first acknowledged one to find a valid block in the longest chain of the network [36]. This type of reward system provides an incentive for miners to contribute their resources to the system. Crypto mining is resource intensive; the more resources a miner invests, the better his chance to solve the puzzle first [36].

Crypto mining can take place on both the **Central processing unit (CPU)** or the **Graphics processing unit (GPU)**. For most popular cryptocurrencies, mining using one's own resources is not profitable due to high electricity and hardware costs unless the mining is carried out using specialized hardware [37]. Such hardware can be an Application-Specific Integrated Circuit, generally optimized to compute a single function or set of related functions.

2.3.3 Cryptojacking

The lucrative potential of crypto mining has led to the exploitation of this methodology through cryptojacking, by which a cybercriminal performs unauthorized crypto mining using the victim's computational power and resources for their own financial gain [9]. This unauthorized mining operation costs extra electricity consumption and dramatically decreases the victim host's computational efficiency. In literature, malware used for this purpose is known as cryptojacking [3].

Cryptojacking is a new emerging attack [15]. Cryptojacking attacks were 4 times higher in 2018 than in 2017 [20], and it is estimated that cryptojacking affected 9% of corporate networks in 2023 [4]. Among ransomware and data breaches, cryptojacking is now foreseen as one of the top common cyber crimes [18]. The banking industry, major commercial websites, government and military servers, online video-sharing platforms, gaming platforms, critical infrastructure resources, and even recently widely popular remote video conferencing/meeting programs have all been the victims of powerful cryptojacking malware campaigns [3].

2.3.3.1 Types of Cryptojacking

Cryptojacking malware can be split into two categories [3]:

- Browser-based cryptojacking
- Host-based cryptojacking

The development of web technologies such as JavaScript and WebAssembly enabled interactive web content, which can access the several computational resources of the victim's device. Browser-based cryptojacking malware uses these web technologies to gain unauthorized access to the victim's system for cryptocurrency mining via web page interactions on the victim's CPU. Browser-based cryptojacking became massively popular with malicious actors in 2017, with services such as Coinhive providing straightforward means to monetize visitor's computational resources via mining [7, 38]. See figure 2.2 describing the lifecycle of browser-based cryptojacking.

Host-based cryptojacking is malware that attackers employ to access the victim host's resources and to make it a miner for the malware owner. Compared to browser-based cryptojacking malware, host-based malware does not access the victim's computation power through a web script. Instead,

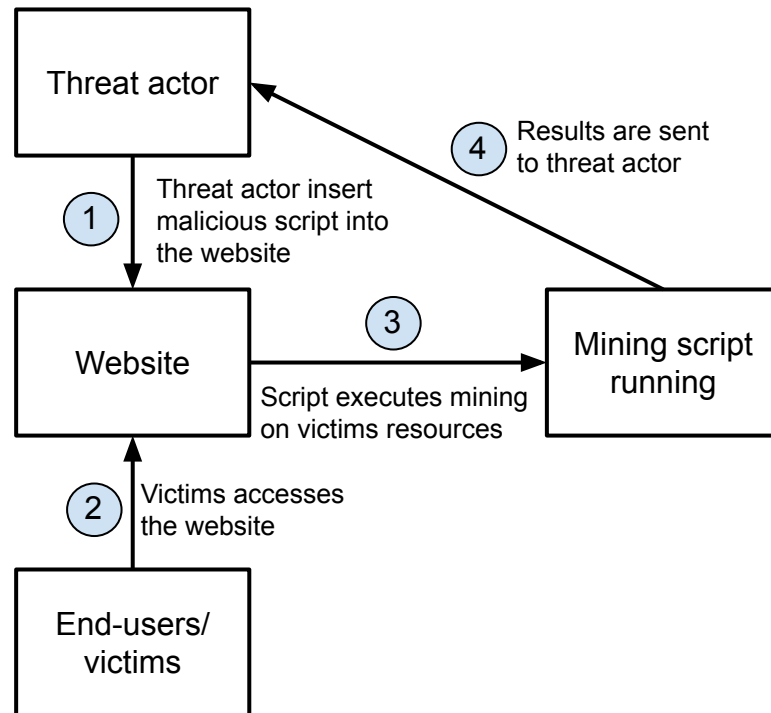


Figure 2.2: Lifecycle of browser-based cryptojacking

they need to be installed on the host system. For host-based cryptojacking, unlike browser-based cryptojacking, where the client directly interacts with the attacker, attackers now need to find a way to deploy and install the malicious mining script on the victim's device [9]. See figure 2.3 describing the lifecycle of host-based cryptojacking.

Most browser-based crypto mining attacks use a modified Coinhive mining script, whereas those running host-based attacks leverage a modified version of XMRig [39]. XMRig is an open-source, cross-platform CPU/GPU miner.

In previous years, attackers focused on browser-based cryptojacking. However, it was noted that the attackers are now shifting their attention to more robust, superior targets, such as cloud servers and cloud infrastructure [38], making host-based cryptojacking malware the new trend [7].

Along with variants of cryptojacking, the attacks are also observed in different infrastructure setups such as the Cloud environment [40]. Cloud-based cryptojacking attacks have been a fast-spreading problem in the last two years [3]. Clouds servers, especially IaaS platforms such as AWS, are being targeted by the attackers because of their host-based capabilities,

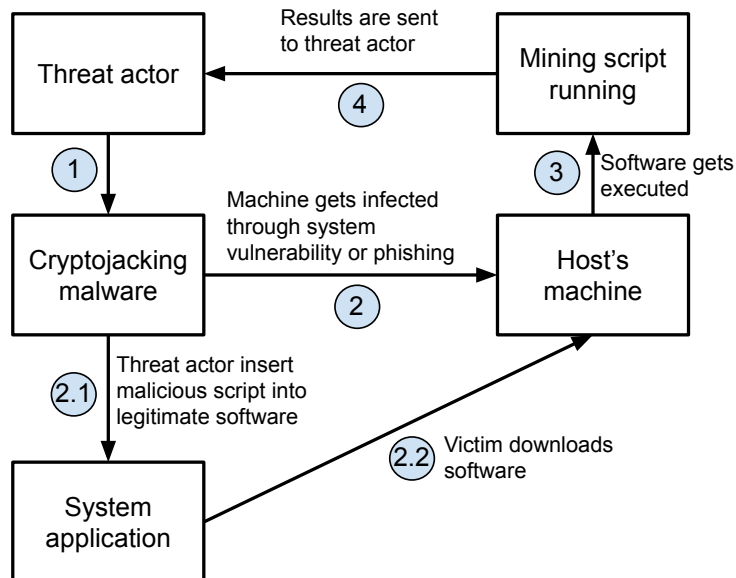


Figure 2.3: Lifecycle of host-based cryptojacking

virtually infinite resources, large attack surface, reliable internet connection and malware spreading capabilities [3]. Cloud servers and infrastructure, often found under-protected and comprising of vulnerabilities, pose as extremely appealing targets for any attacker searching for robust and unrestrained CPU power. For cryptojacking attackers, the cloud is a goldmine [38]. The cloud's complex nature means it's easy for crypto miners to go undetected for a considerable time.

2.4 Malware Detection

Malware detection determines if the content of a file is malicious or not. Any software which intentionally executes malicious payloads on victim machines is considered malware [8]. To protect computer systems and the internet from malware, the malware must be detected before it affects many systems.

The number of malware is increasing at an alarming rate, and some malware can hide in the system by using different obfuscation techniques [8]. Obfuscation is a strategy to hide the true intent and functionality of a program, and common methods are compression, encryption, and encoding. Most modern malware uses obfuscation and evasion techniques to avoid detection

[41].

In the traditional malware detection literature, there are two main analysis methods: static and dynamic [3]. Static malware detection relies on features extracted from executable artefacts such as opcodes, bytecodes, or strings, while dynamic techniques are based on behavioural features from system calls, captured network traffic, etc [42]. In other words, static analysis examines the malware without running the actual code, while dynamic analysis examines the malware's behaviours while running its code. No approach can detect all malware in the wild. Static methods are fast and efficient in detecting known malware but have problems detecting new or obfuscated malware. Dynamic methods detect new and different malware variants more effectively but are more complex and time-consuming to use [8].

Signature-based detection is a static approach widely used within antivirus and malware analysis software [41]. However, sole reliance on signature-based security falls short in combating today's cyber threats [43]. This happens because of the very fast spread of computer malware and the increasing number of signatures [44]. Still, online scan engines such as VirusTotal [45] are heavily used by researchers to label malicious URLs and files, and many recent works rely on VirusTotal's API for data labelling [46]. VirusTotal is a meta-antivirus comprising more than 60 security vendors to aggregate their scanning results. In practice, if a certain number of vendors label a file/URL as malicious, one would label the file/URL as malicious.

2.4.1 Malware Packing

Recently, a trend of increasing complexity and obfuscation has been observed in malware. Packing executable files is one of the most common techniques for code protection and has been repurposed for code obfuscation by malware authors. A packer is a program that transforms an executable binary file into another form using compression and/or encryption to protect or hide the executable's original content. Packing is also malware authors' most common obfuscation method to disrupt malware detection and analysis. Packed files can be identified using entropy analysis. Entropy analysis measures the uncertainty in an executable's data without prior knowledge of the executable. Both encryption and compression transform a one-byte sequence into another, which can cause an increase in the entropy of their input. Thus, high entropy may disclose the presence of packed malware [39]. Most packers are available free of cost and are easy to use. Attackers generally use well-known packers such as **Ultimate Packer for eXecutables (UPX)** [3] for binary obfuscation.

2.4.2 Detecting Cryptojacking

There is only a limited amount of research on detecting cryptojacking [7]. Given the prevalent emerging nature of the cryptojacking malware, it is vital to detect and prevent unauthorized mining operations from abusing any computational resources without the users' consent [3]. Cryptojacking is known to use successful and complicated evasion techniques to circumvent security measures [38]. Existing detection methods, such as browser extensions that protect users with blacklist methods or antivirus programs with different analysis methods, can only provide a partial solution to this emerging cryptojacking issue. Attackers can easily bypass these methods using obfuscation techniques or frequently changing their domains or scripts. Therefore, many studies in the literature proposed cryptojacking malware detection methods using various dynamic or behavioural features [3]. Most of the research conducted on cryptojacking detection has focused on browser-based cryptojacking malware [7].

Though it is critical, detecting cryptojacking is challenging because it differs from traditional malware in several ways. First, it abuses the victim's computational power instead of harming or controlling it, as in the case of traditional malware. One of the most noticeable characteristics of a crypto miner is the CPU utilization it affects [38]. Traditional malware detection and prevention systems are optimized for detecting the harmful behaviours of malware, but cryptojacking malware only uses computing resources and sends back the calculated hash values to the attacker. Therefore, crypto mining is often considered by the system to be a heavy application that needs high-performance usage. Second, crypto miners can also be used or embedded in legitimate websites, which makes them harder to notice because those websites are often trustworthy, and users do not expect any unauthorized mining on their computers. Third, whereas traditional malware attacks typically aim to retrieve sensitive information by gaining control of the victim's machine, in cryptojacking attacks and other exfiltration attacks, the attacker's primary objective is to maintain undetected within the system for as long as possible. The cryptojacking attack's revenue is directly proportional to the time a cryptojacking malware goes undetected. Therefore, attackers use filtering and obfuscation techniques that make their malware harder for detection systems and harder to be noticed by the users [3].

Still, antivirus software is commonly used against host-based cryptojacking malware [3]. Recent studies also show the relevance of VirusTotal as a valid and practical cryptojacking meta-detector [47].

2.4.3 Yet Another Recursive Acronym

A recent and flexible solution for signature-based malware detection is **Yet Another Recursive Acronym (YARA)**, which enables the creation of rules to identify and classify malware based on patterns of binary and textual data [41]. **YARA** is an open-source tool, and using it has already become an established technique in malware analysis [48]. The rules are formulated through the reverse engineering of malware families to identify **Indicator of Compromises (IoCs)**.

YARA rules consist of three sections: meta, strings and condition, as shown in the example rule in Listing 2.1. Meta provides information about the rule, such as the author, the date it was created, the description, etc. The string section is where you define the patterns, signatures or strings the rule will look for. These are called **IoCs** and can be classified into three types of strings: text strings, hexadecimal strings and regular expression strings. The condition section is the only required section and specifies when the **YARA** rule is valid for the input. It uses a Boolean expression to determine the result of the scanning.

Listing 2.1: Example of a YARA-rule

```
rule exampleRule {
  meta:
    description = "Detect presence of 'Hello'
                  or hex of 'world'"
    author = "Edvin Baggman"
    date = "2024-05-02"

  strings:
    $a = "Hello"
    $b = { 77 6F 72 6C 64 } // Hex for 'world'

  condition:
    $a or $b
}
```

YARA rules are effective due to their customisable features by which anyone can develop their own rules [48]. However, developing high-quality rules to detect a malware family of interest can be time intensive because of the tasks necessary to build good signatures (e.g. reverse engineering a malware) [49]. Weak rules will result in malware evading detection or false positives.

YARA rules are also effective at detecting obfuscated malware, as many rules are built to detect different obfuscation capabilities, such as packing [41]. Also, **YARA** rules can identify capabilities in new malware, such as including encoded sections.

2.4.3.1 Evaluating YARA Rules

To evaluate **YARA** rules, often standard evaluation metrics for classification models are used [50]. This includes using Accuracy, Precision, Recall and F1-Score to measure how well the rule performs (i.e. if it detects the malware family it is supposed to). Accuracy measures the overall correctness of the model and is defined as the ratio of correct predictions. Precision measures the accuracy of positive predictions. Recall indicates the proportion of actual positives that were correctly identified. The F1 Score is the harmonic mean of precision and recall. It helps evaluate a rule's effectiveness at catching as much malware as possible (high recall) while minimizing false alarms (high precision). To evaluate the efficiency of a rule, a balance of Precision and Recall is important [50].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

2.5 Related work area

Below is a summary of previous research related to this work.

2.5.1 Kelly et al.

An analysis of honeypots on different cloud platforms was made in 2021 [21]. The article investigates various types of attacks on **IaaS** cloud environments like **GCP**, **AWS**, and **Azure**. With the use of T-pot [51], they deployed a variety of low and medium-interaction honeypots using docker containers on every cloud provider. A total of 9 deployments of T-pot were made. The

study provides insights into attackers' activities on honeypots deployed on these cloud providers, analyzing attack volumes, targeted services, credentials, vulnerabilities, and threat intelligence feeds. It also offers a region-based analysis of attackers' activities across North America, Asia, and Europe.

The study suggests discrepancies in the attacker's activities across cloud service providers. In the context of the study, an attack refers to any unauthorized attempt or activity directed towards the honeypot. These attacks include various actions such as probing, scanning, or exploitation attempts. **GCP** received far more attacks overall, with most honeypots receiving 10–20 times more attacks on **GCP** than on **AWS**. Specifically, on **GCP**, Cowrie received 297,818 attacks during a 3-week period, while only receiving 4503 attacks on **AWS** in the same period. On **Azure**, Cowrie received 9012 attacks.

The analysis of attacker source IPs in **GCP** revealed that China ranked first with over 25% of the attacks, followed by the United States with 20% of attacks. Vietnam, which ranked first in **AWS** honeypots, ranked fourth after Russia in **GCP**. The geolocation of attacker source IPs showed dominance from America, Asia, and Europe, with a significantly higher contribution from the East Coast of the United States compared to **AWS** results.

Interestingly, the credentials used (usernames and passwords) to access Cowrie differed between the cloud service providers. The attackers in **GCP** predominantly used default credentials and focused on system information gathering and privilege escalation.

The analysis of vulnerabilities targeted by the attackers indicated a range of known vulnerabilities, suggesting attempts to exploit vulnerabilities dating back to 1999. The operating systems of attacking devices varied across regions, with Windows being the most frequently used.

2.5.2 Jayasinghe and Poravi

Keshani Jayasinghe and Guhanathan Poravi [38] documented attack instances of cryptojacking targeting cloud infrastructure. They discuss the shift of cryptojacking attacks from browser-based targets to cloud infrastructure. It analyzes 11 practical attack instances targeting cloud servers and infrastructure, comparing their properties with the limitations of existing literature on detection systems. The paper outlines the benefits for attackers and adverse effects for victims of cryptojacking in cloud environments.

Benefits of Cryptojacking in Cloud for Attackers:

- Considerably greater **CPU** resources for a bigger payoff

- Larger attack surface
- Spreading capabilities within the network
- Accessing data and credentials in cloud servers
- Ability to backdoor the server and leave it more vulnerable to future attacks
- More reliable than depending on a network of browsers

Adverse Effects of Cryptojacking in Cloud for Victims:

- Substantial damage to the servers
- Large resource consumption bills
- Device performance, degradation, and malfunction
- High-priority organizational tasks and real processes cease to operate or are lowered in priority

The study observed **OS** differences, where most attacks targeted Windows. 60% of the attacks targeted Windows, and 26,7% of the attacks targeted Linux. While examining the attack details, it was noted that the majority of the attacks have made use of the Monero **CPU** miner, XMRig. A large number of attacks made use of a fileless infection technique. These cryptojacking attacks were not seen as writing an executable directly to the victim's device. In its place, the payload will, in most cases, acquire the mining script using PowerShell.

2.5.3 Zimba et al.

An investigation of state-of-the-art crypto mining attacks was made [52]. The paper examines the two most common attack approaches: browser-based and host-based cryptojacking attacks. The investigation involved analyzing the attacks through both static analyses of the malware code and dynamic analysis of its behaviour. The paper also presented some **IoCs** for cryptojacking attacks to help mitigate attacks.

The results from the study showed that Monero is the most commonly mined cryptocurrency in these attacks since the mining thereof does not require specialized hardware. Cybercriminals implant crypto-mining malware into legitimate software to make the attack effective, thus acting as trojans.

One of the most observed utilities that is covertly integrated with legitimate software is XMRig. The most common attack approach involves installing and running the malware executable in memory, usually downloaded upon infection. The crypto-mining malware was observed to employ detection evasion techniques in an effort to avoid detection.

2.5.4 Baser et al.

This study by Melike Baser, Ebu Yusuf Guven, and Muhammed Ali Aydin focuses on the analysis of malicious files gathering via a honeypot and benchmarking of antivirus software [43]. The study highlights the evolving nature of malware threats and the importance of effective detection methods in combating cyber threats. Key points include signature-based detection in antivirus applications and the challenges of obtaining timely responses from antivirus databases. The study emphasizes the limitations of relying solely on signature-based detection and suggests the importance of incorporating additional analysis techniques for comprehensive malware detection.

They deployed the Cowrie Honeypot to **GCP** and analyzed suspicious files uploaded by attackers through a signature-based method using 64 antivirus programs. The honeypot was deployed for 47 days, and from the attacks on the Cowrie honeypot, 50 suspicious files were retrieved. In their experiments, only three of the antivirus programs exhibited success rates exceeding 90%. The majority, 61 antivirus programs, showed success rates predominantly below 70%. AVG was the anti-virus application with the highest success rate, followed by Avast and DrWeb applications. The antivirus programs tagged the files with different classifications. The most common classification was Mirai, the second was Trojan, and the third place was CoinMiner, which corresponds to a cryptojacking attack.

Chapter 3

Method

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 focuses on the data collection techniques used for this research. Section 3.3 and Section 3.4 describe the method used for the analysis, while Section 3.5 describes how the analysis can help in detecting cryptojacking. Finally, Section 3.6 informs about some ethical considerations.

3.1 Research Process

Figure 3.1 illustrates the overall research process. First, the pre-study was conducted, resulting in method choices for the thesis. These are a) selecting a suitable honeypot and b) selecting a suitable malware detection solution. Second, honeypots were deployed to multiple cloud service providers. Third, the honeypot logs and downloaded files were extracted and analyzed using different techniques. Finally, the results from the analysis of the potential cryptojacking malware were used to aid in detecting cryptojacking attacks.

3.2 Data Collection

To answer the research questions, up-to-date data needs to be collected and analyzed. To obtain this data, Cowrie, in medium interaction mode, was deployed to multiple cloud service providers.

Cowrie emulates a Unix system, meaning that most attacks can be expected to target Unix systems. Consequently, the downloaded executable files are expected to be in the **Executable and Linkable Format (ELF)** format, a

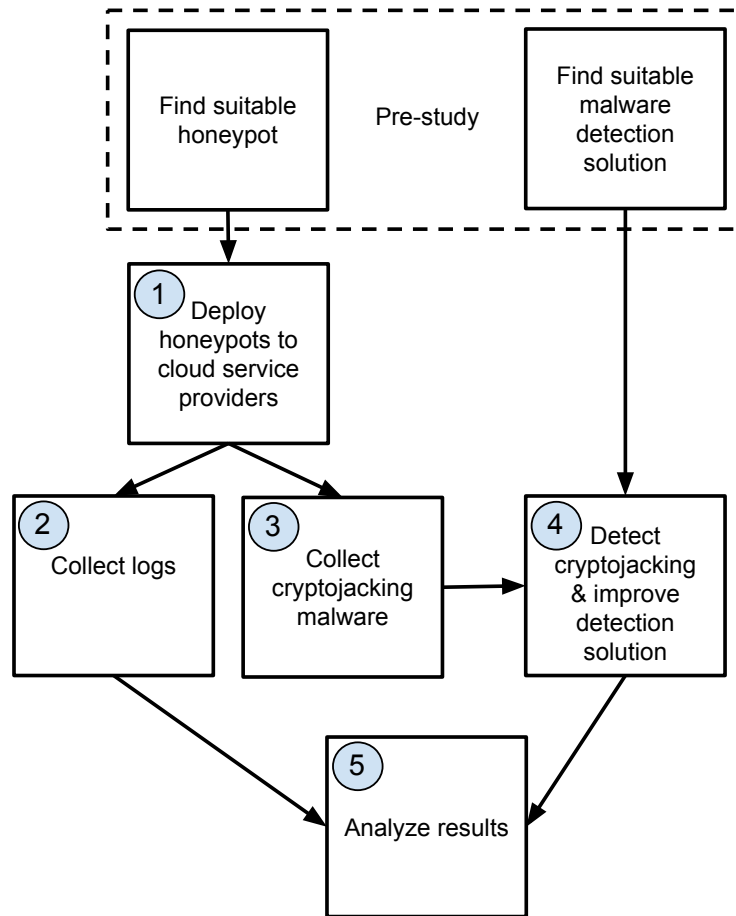


Figure 3.1: Research Process

commonly used format for binaries in Unix-based systems. Cowrie collects data in JSON format, and every downloaded or uploaded file is securely stored where attackers cannot execute them. The result will be one JSON file for each honeypot per day and a directory with downloaded files for each honeypot.

Cowrie was the selected honeypot due to its simplicity, active development, and data collection, which aligns with the study's goals. Another option could be to deploy the T-Pot all-in-one honeypot platform, which deploys 20+ honeypots and has multiple visualization options. However, a lot of the data collected would be redundant, and to focus the research on the research questions, only the Cowrie honeypot is used.

3.2.1 Cowrie Deployment

The Cowrie honeypot was deployed to **GCP**, **AWS** and **Azure** for a total of 4 weeks, divided into 2 rounds for 2 weeks each. The first period was between 2024-02-26 and 2024-03-11, while the second was between 2024-03-11 and 2024-03-25.

These cloud service providers were chosen because they are the industry's three largest and most utilized platforms, providing a higher chance of observing real-world attack behaviours and trends. The 4-week time period was selected to provide sufficient duration to spot patterns and still leave time for analyzing the data within the project's time frame.

3.2.1.1 Virtual Machine Configuration

The **VMs** were set up to be as similar as possible. See 3.1 for details on the configurations. The zones were chosen to be as geographically close to each other as possible and inside Europe to comply with the legal issues described in Section 2.2.2. Since Cowrie doesn't require significant computing power to operate, a weak/cheap type can be chosen. The installation guide for Cowrie suggests the Debian **OS**. Note that the specifications displayed to the attacker inside the Cowrie honeypot depend on specific Cowrie configurations, not the machine's configuration on which it is deployed.

Table 3.1: Configurations used on virtual machines

Provider	Zone	Type	OS
GCP	europe-north1-a	e2-medium	Debian 12
AWS	eu-north-1b	t3.micro	Debian 12
Azure	North Europe (Zone 1)	Standard B1s	Debian 11

It is also important to ensure that the firewall accepts connections to the ports on which Cowrie operates. Firewall rules were set up to allow all ingress connections on port 22 and 23.

3.2.1.2 Cowrie Configurations

Following the Cowrie installation guide, there are a couple of optional steps. The `cowrie.cfg` file can be modified to set specific values of the honeypot. See Listing 3.1 for specific configurations used for all deployments in this study. Host name and address are set to make it consistent between cloud

service providers. SSH version is set to an old version that has known vulnerabilities. A more convincing environment is set by replicating a genuine kernel version and build string commonly found in Debian systems.

Listing 3.1: Configurations used on the Cowrie honeypot (cowrie.cfg)

```
[honeypot]
hostname = dev
fake_addr = 192.168.120.214

[telnet]
enabled = true

[ssh]
version = SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1

[shell]
kernel_version = 4.6.0-nix-amd64
kernel_build_string = #1 SMP Debian 4.6.4-1 nix1
```

Furthermore, Cowrie's listening ports were also changed to the default SSH and Telnet ports, 22 and 23, respectively, using iptables port redirection. In the default configuration, the listening ports would be 2222 and 2223.

3.2.1.3 Cowrie Honeypot Obscurer

To make it harder to fingerprint the deployments as honeypots, the tool Cowrie Honeypot Obscurer [31] was used. However, some modifications to the tool were made to keep the configurations consistent between the deployments and ensure attackers could access the honeypot with relative ease. These changes include:

- Only 1 option for processor: Intel(R) Core(TM) i5-4590S CPU @ 3.00GHz
- Only 1 option for user: admin
- Any combination of username and password will grant access to the system

3.2.2 Validity and Reliability of Data

The data collected from the Cowrie Honeypot is the logs from brute force attacks and the shell interaction performed via SSH and Telnet. This means the honeypot has a limited field of view, as it only records attacks on ports 22 (SSH) and 23 (Telnet). Also, honeypots risk being fingerprinted, which can lead to attackers cancelling their attacks. These issues make the data collected not entirely translatable for what one could expect on a production system.

The ideal method for this project would be to use real vulnerable systems in the cloud instead of a honeypot. This would increase the validity of the data collected since it would be impossible to fingerprint the system as a decoy system. However, the amount of work needed to monitor and clean the systems makes this approach untenable, which is why honeypots were built in the first place. Still, the data collected by the Cowrie honeypot is a valuable asset in understanding threats. Even though it might not give an entirely correct representation of a threat landscape, honeypots are still used in research to map out current threats, as described in Section 2.2.

The threat landscape is continuously evolving with new and improved attacks. Therefore, the data collected is tied to the specific time frame in which the project was performed. If someone were to perform the exact same data collection method, the data collected would likely look different.

3.3 Methods for Analysing Honeypot Logs

The deployment of honeypots creates log records describing the incidents that have occurred. The following section describes some metrics used for honeypot data and the process of determining them. For more information about honeypot data analysis, see [19].

Unique sessions are important when analyzing honeypot data. A new session is created every time a new interaction with the honeypot is initiated. For instance, a malicious actor trying to connect to the honeypot at two different times creates two sessions. A malicious actor entering five commands inside the honeypot is still one session.

3.3.1 Attack Source

If an attack occurs, one could want to specify where the attacks came from. This can be done in different granularities, such as Country, IP address, OS. As stated in section 2.2.2, any published data of research honeypots needs to be

anonymized. Therefore, no IP address is exposed. In this study, the originating country extracted from the IP address and the overlap of attacking IP addresses is shown.

3.3.2 Attack Frequency

One of the fundamental questions while deploying honeypots is how often the honeypot is attacked. The following metrics can be used to define attack frequency:

- Number of attacks per time unit
- Number of sources per time unit
- Number of attacks per source
- Number of sessions per time unit
- Number of downloaded payloads per time unit

3.3.3 Attack Evolution

To understand how attacks evolve, one would want to learn what normal behaviour is and to spot if this normal behaviour changes. A time evolution model created by linear regression can be used to investigate if attacks can be estimated from the combination of the previous attacks. The linear regression model is defined by:

$$Y^*(i) = \sum \alpha_j X_j(t) + \beta, \quad j = 1, 2, \dots, k$$

where $X_j(t)$ denotes observed attacks from honeypot/category j at time t , α_j and β are best fit linear model parameters.

The correlation factor measures the correlation between models and can be used to rate whether the observed events are expected or vary substantially.

Using linear regression, one can also examine whether a model based on observations restricted to a specific cloud service provider can reliably describe observations from other cloud service providers or whether the observations vary substantially.

3.3.4 Propagation of Attacks

In addition to examining attacks individually, efforts should also focus on identifying the propagation of attacks across multiple honeypots. Propagation takes place when one attacking IP address is observed on one platform and then subsequently on another. Propagation of attacks can be visualized in a propagation graph.

3.3.5 Attack Patterns

A technique to analyze honeypot data is to detect common time series patterns. To find these patterns, attack series can be categorized into classes of activity. A study about attack pattern discovery in honeypot data [53] showed the attacks can confidently be grouped into three classes of activity.

- Continuous activity
- Sustained bursts
- Ephemeral spikes

This can be done using **Symbolic aggregate approximation (SAX)**, which approximates time series data by segmenting into time intervals of equal size and summarizing each of these intervals by its mean value. Each interval is then mapped to a finite set of symbols, such as the categories above. Binning continuous data into intervals can be seen as an approximation that reduces noise and captures the trend of a time series.

SAX bins are calculated using the percent point function, which is the inverse of the cumulative distribution function. This approach ensures that each bin represents an equal proportion of the data when the data is normalized. However, they are denormalised to make these bins easier to interpret in this context. Denormalizing the bins involves adjusting them back to the original scale of the data using the mean and the standard deviation of the data set.

3.4 Methods for Malware Analysis

The objective of malware analysis is to understand the features of the malware by examining both the internal program logic and its behaviours. An approach for static malware analysis was presented in [52] and contains 5 stages

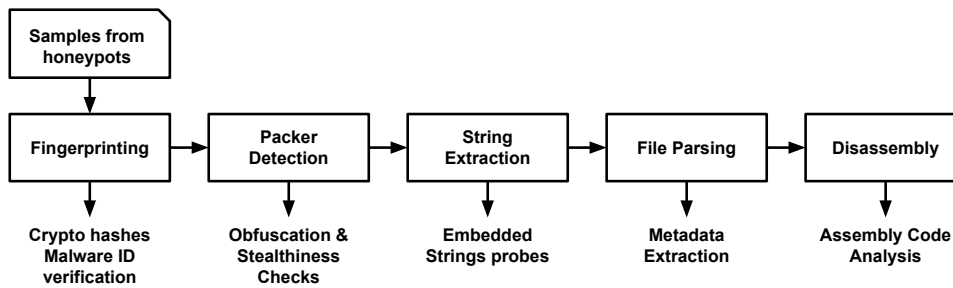


Figure 3.2: Static malware analysis process flow

explained in the coming sections. See figure 3.2 for an overview of the static malware analysis process flow.

There are a lot of tools that can be used for malware analysis. In this study, the tool **Detect It Easy (DIE)** [54] was used to detect packing in Section 3.4.2, string extraction in 3.4.3 and metadata extraction in 3.4.4. The tool Ghidra [55] was used for disassembly in Section 3.4.5. Both **DIE** and Ghidra are free and open-source software.

3.4.1 Fingerprinting

Fingerprinting involves generating the cryptographic hash values for the suspect binary based on its file content. The hashes are then used in reputed malware databases. In this study, VirusTotal is used as the malware database. Aggregating the labels from the engines in VirusTotal provides good accuracy in labelling malicious files [56]. Note that the classifications in VirusTotal are not definitive since the antivirus vendors often update the signatures used for their classification. The methodology works as follows: If t or more engines give a malicious label, the file is marked as malicious.

Without obfuscated benign files in the dataset, t can be set to a low number, between 2–15 [56]. VirusTotal can, therefore, be used as a cryptojacking oracle, where the oracle returns labelling feedback for the binary variant: *cryptojacking malware* or *not cryptojacking malware*. Assuming that no files collected in the honeypot are benign, we can set t between 2 and 15. For this study, the following was used as a cryptojacking oracle:

If 7 or more engines in VirusTotal gives a *miner* label, the file is marked as *cryptojacking malware*, otherwise we mark the file as *not cryptojacking malware*.

3.4.2 Packer Detection

This step involves looking closer at the malware to detect if a packer has been used on the file in an effort to disguise its internal logic. Using a tool to look at the file's entropy, information about whether the file is packed can be gained. Also, a packed file often includes the string of the packer used somewhere in the binary. Therefore, if and which packer has been used can usually be detected. The packed files must be unpacked to analyse the malware samples further.

3.4.3 String Extraction

Strings are sequences of characters embedded within a file. An effective static malware analysis approach is to extract strings to give clues about the functionalities associated with a suspect binary. Strings extracted from the binary can reference filenames, URLs, domain names, IP addresses, attack commands, registry keys, etc. Although strings do not give a complete picture of the purpose and capability of a file, they can give a hint about what malware is capable of doing. Since most of the strings that can be extracted from malicious code are composed of information closely related to malicious behaviour, extracting embedded strings related to crypto mining can give information about whether the software is used for crypto mining.

3.4.4 File Parsing

Extracting metadata from the files is used to gain insights into the functionality and origin of the malware. This involves analyzing the code to identify and extract various information that is typically not visible through simple observation. Metadata extracted from the binary can reference the intended operating system, architecture, linking type, compiler, and language.

3.4.5 Disassembly

Most malware is distributed in compiled binary form, making it unreadable by humans directly. When dealing with these binaries for which source code is not available, disassembling them can give important information about the malware. Disassembly is used to reverse engineer a binary and to understand its functionality, techniques, and behaviour. In short, by examining the disassembled code, you can identify what the malware is designed to do. Moreover, through disassembly, specific patterns or indicators of the malware

family can be discovered that can be used to detect and mitigate the malware across systems.

3.5 Mitigation of Cryptojacking Attacks

Having identified and analysed cryptojacking malware, the next step would be to leverage the gathered information into something usable to detect and mitigate such attacks. **YARA**, a tool designed for classifying and identifying malware families, was used in this study.

Anyone can develop rules for **YARA**, thereby contributing to identifying and mitigating the detected malware for the broader community. Utilising **YARA** in conjunction with honeypots enables the translation of intelligence gathered from honeypots into practical, usable applications.

3.5.1 Development of YARA Rules

The construction of **YARA** rules requires understanding malware signatures and behaviours that are distinct and constant across different instances. The resulting **YARA** rules in this study should minimize false positives by avoiding generic or common patterns and ensuring the detection of new and slightly modified variants of known malware families. The rules' development process involved identifying patterns from the malware analysis phase.

3.5.2 Evaluation of YARA Rules

The **YARA** rules were evaluated using the four standard evaluation metrics for classification models: Accuracy, Precision, Recall and F1-Score.

To evaluate the **YARA** rules, a controlled set of cryptojacking malware needs to be constructed, for instance, from online malware databases. This study used the malware database MalwareBazaar [57]. Here, people can upload their malware samples collected from-the-wild, and tag them with a list of traits describing the malware. Each malware was manually analysed using the method described in Section 3.4 and placed in relevant categories.

3.6 Ethical Considerations

The study has a potential ethical concern: the possibility that the research could inadvertently aid malicious actors in their cryptojacking attempts by

revealing specific detection methods that could be circumvented. However, the positive aspects of providing up-to-date results about the understudied subject of host-based cryptojacking will hopefully outweigh these risks.

Additionally, it is hard to understand the full functionality of malware. Without knowledge of potentially malicious activities tied to the cryptojacking malware, it is difficult to justify purposely executing the malware or to make a dynamic analysis of the malware. Therefore, considerations must be taken when handling the files caught in the honeypots to avoid exposing active malware to the surrounding environment.

Chapter 4

Results and Analysis

In this chapter, the results are presented. They are divided into three main parts: an analysis of the honeypot logs (Section 4.1), a malware analysis (Section 4.2) and the development of YARA rules (Section 4.3). Section 4.4 contains an analysis of the validity of the results.

4.1 Analysis of Honeypot Logs

In total, 488 MB of JSON logs have been collected over the time the honeypots were deployed. An entry in the logs could look like this:

Listing 4.1: Entry in cowrie.json describing the occurred incident

```
{
  "eventid": "cowrie.command.input",
  "input": "uname -s -v -n -r -m",
  "message": "CMD: uname -s -v -n -r -m",
  "sensor": "VIRTUAL-MACHINE",
  "timestamp": "2024-03-04T01:13:34.942593Z",
  "src_ip": "IP-TO-ATTACKER",
  "session": "da803a3fdbad"
}
```

4.1.1 Attack Source

Identifying an attacker can be done in different granularities. As described in Chapter 3, only the origin country of the attack and the overlap of attacking IP addresses will be considered in attack sources. In figure 4.1, the distribution of attacking countries on the different cloud service providers is presented.

China is undeniably the country from which most attacks originate, followed by the United States.

Interestingly, the origin country of attack seems to differentiate between the cloud service providers. For example, **AWS** has 113 more unique IP addresses attacking from India than from South Korea, while South Korea has more unique attacking IP addresses than India on **GCP** and **Azure**.

In figure 4.2, we can see the overlap of attacking IP addresses. Out of the 13367 total unique IP addresses observed, only 186 were seen on all three cloud service providers.

It is worth noting that an attacker can use a **Virtual private network (VPN)** or a proxy server to make it look like the attack is coming from a different IP than their own. Still, the results are valuable as they can help identify attack patterns and trends and help understand how attackers operate.

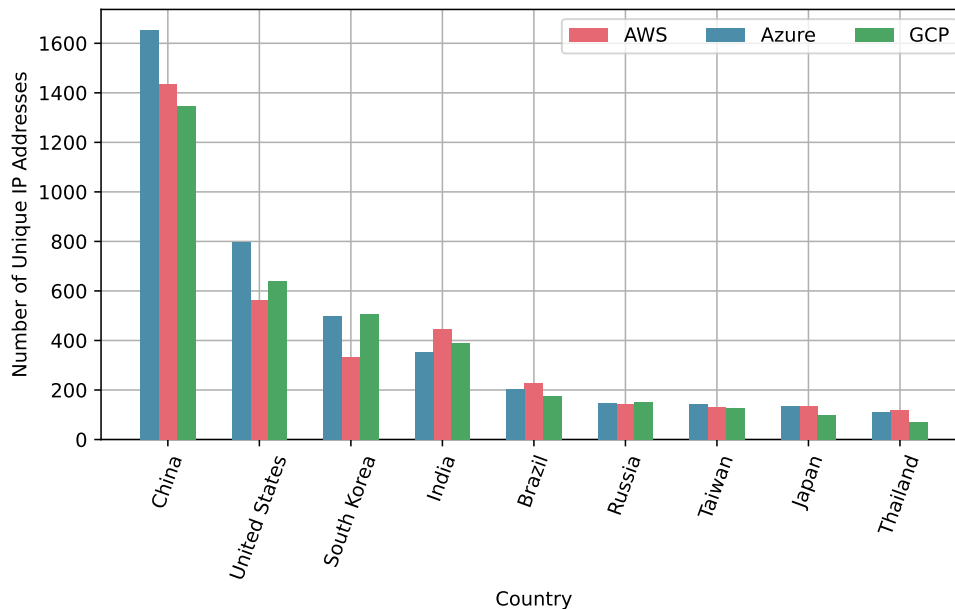


Figure 4.1: Origin of unique IP addresses attacking the cloud service providers

4.1.2 Attack Frequency

The honeypots are all attacked within minutes after they have been deployed. After deployment, the honeypots were continuously targeted, where the longest time in between attacks was recorded on **GCP** and was 115 minutes. On **AWS** the longest time between attacks were 83 minutes, and on **Azure**

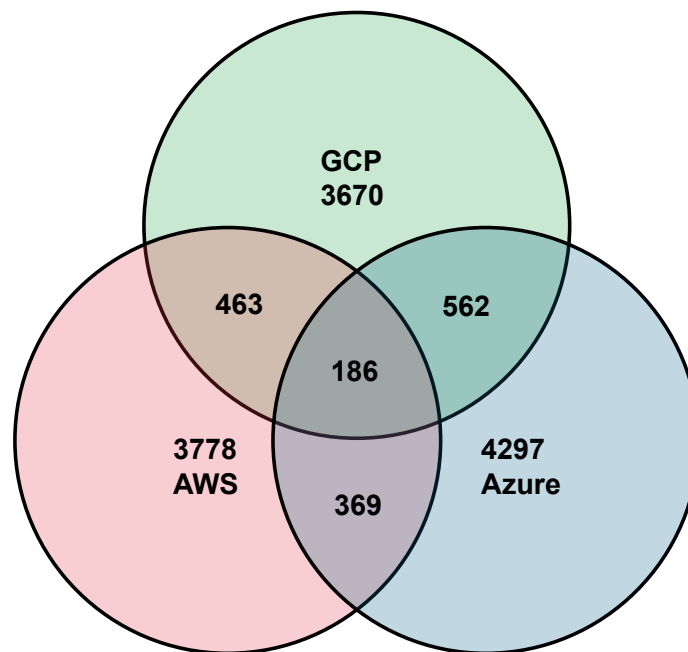


Figure 4.2: Venn diagram of unique IP addresses observed attacking the cloud service providers

it was 43 minutes. On average, the time between attacks was 43 seconds across all platforms. As described in Section 2.1.1, the VMs are assigned an IP address from the cloud service providers' public IP ranges. They are not published anywhere.

As stated in Section 3.3, a new session is created every time a new interaction with the honeypot is initiated, not for every command executed. Each sessions is considered as an unique attack towards the honeypot.

In table 4.1, you can see the total number of attacks on the three platforms. In table 4.2, the same data is presented, but as an average per day. In table 4.3, you can see statistics about downloaded files per command executed, commands executed per session and number of sessions per unique IP address. Interestingly, sessions per IP vary quite a lot between the platforms, while the other statistics remain similar.

Table 4.1: Total number of sessions, commands executed, unique IP addresses and downloaded files on the platforms

	Attacks (Sessions)	Commands	IP addresses	Downloaded files
GCP	51 345	46 937	4881	104
AWS	37 378	34 073	4796	62
Azure	79 543	73 024	5414	128
Total	168 266	154 034	15091	294

Table 4.2: Number of sessions, commands executed, unique IP addresses and downloaded files per day on the platforms

	Attacks (Sessions) per day	Commands per day	IP addresses per day	Downloaded files per day
GCP	1834	1676	174.3	3.7
AWS	1335	1217	171.3	2.2
Azure	2841	2608	193.4	4.6
Total	6010	5501	539	10.5

Table 4.3: Number of downloaded files per session, commands executed per session and sessions per IP address on the different platforms

	Downloaded files per Session	Commands per Session	Sessions per IP
GCP	0,0020	0,91	10,5
AWS	0,0017	0,91	7,8
Azure	0,0016	0,92	14,7

4.1.3 Attack Evolution

No correlation could be found between the observed events from different time intervals or between observed events from different cloud service providers. When estimating the events in a different time interval or with another cloud service provider, only R2-values below 0.02 could be found, indicating a very low correlation.

However, one observation with a high correlation factor was found. This observation estimated all attacks on **GCP** using the attacks on **GCP** from the United States. The estimated model obtained was defined by:

$$Y^*(t) = 0.975X_j(t) + 882.65$$

$X_j(t)$ represents the evolution of the number of attacks from the United States on **GCP**. The R2-value for this model is 0.933, indicating a strong correlation. Plotting the estimated model, it is clear that one single day with many attacks is responsible for most of the correlation and that the model is not a good estimation.

4.1.4 Propagation of Attacks

In this section, scenarios are analyzed where propagation between two honeypots is assumed to occur when the IP address of an attacking machine observed at one honeypot is previously observed at another. In the propagation graphs, each node is a honeypot, and the transition between two nodes identifies a propagation between the nodes. A probability is associated with each transition, explaining the likelihood of an observed IP address later being observed at the other node.

Figure 4.3 presents the propagation graphs between the cloud service providers, considering the attacks from the entire 4-week period and with the period split. Looking at the graph considering attacks from the entire period, we can see that an observed IP address on **Azure** is more likely to be later observed on **GCP** than on **AWS**. We can also see that an IP address observed on **GCP** and **Azure** has a much higher chance to be observed again at itself than the same occurring on **AWS**. Considering the graph with the periods split, we can see that an observed IP address on **AWS** is more likely to be later observed on the other providers than on itself.

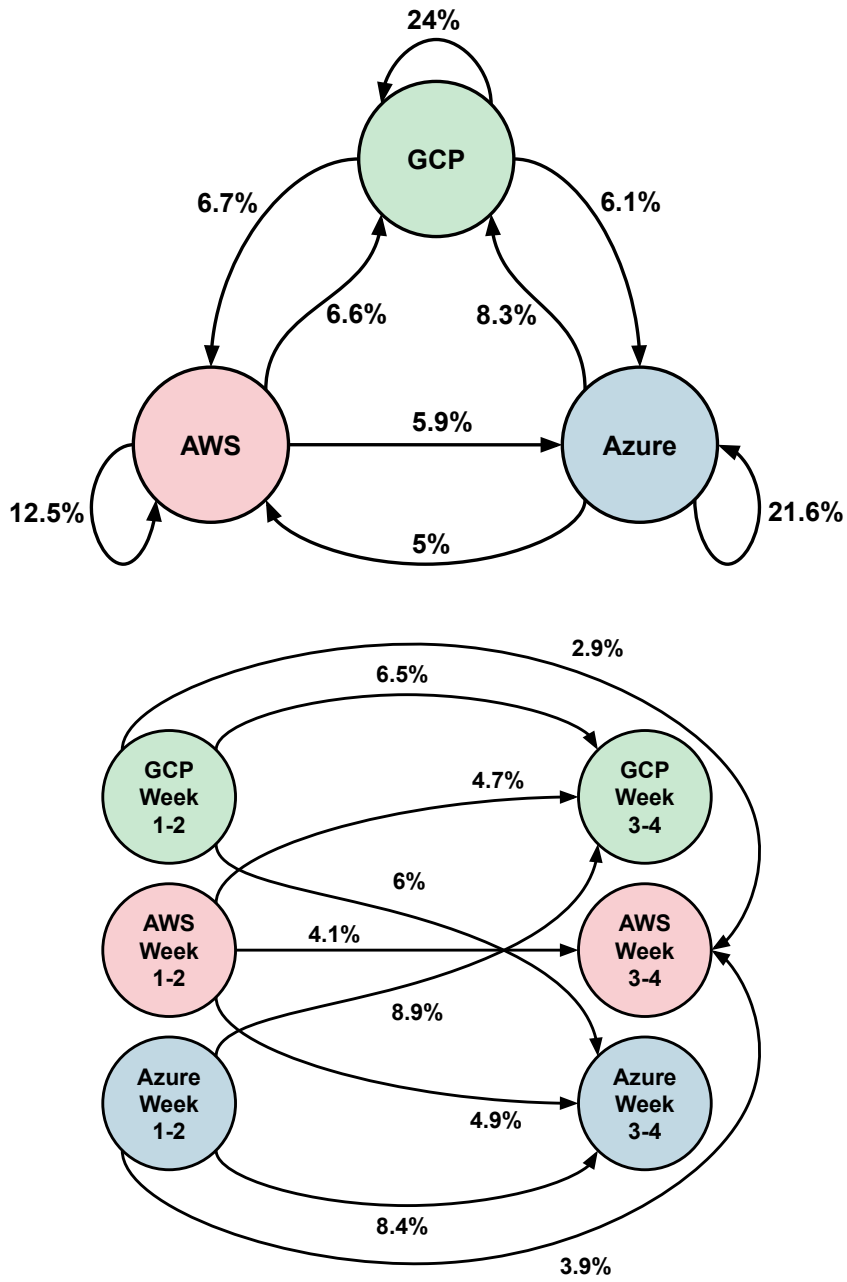


Figure 4.3: Propagation Graph over the cloud service providers. The graph shows the likelihood of an observed IP address later being observed at the target honeypot. In the first graph, the whole 4-week period is used. In the second graph, the periods are split.

4.1.5 Attack Patterns

The number of unique sessions is analyzed to find attack patterns. The 4-week period is split into 112 chunks of 6 hours each, where the total number of unique sessions in that time frame is the value for the chunk. The size of the chunks can be chosen arbitrarily depending on the data, and similar results have been observed when considering different granularities of the chunks.

The mean value for unique sessions per chunk is 500.47, and the standard deviation is 1790.93. The breakpoints for the bins are described in Table 4.4. The results are visualized in a combined graph in Figure 4.4 and the results from **SAX** are described in numbers in Table 4.5. The results show that **Azure** has more sustained bursts than the other providers.

Table 4.4: Breakpoints for bins used in **SAX**

No activity	Continuous activity	Sustained bursts	Ephemeral spikes
$sessions = 0$	$0 < sessions \leq 500$	$500 < sessions \leq 1707$	$1707 < sessions$

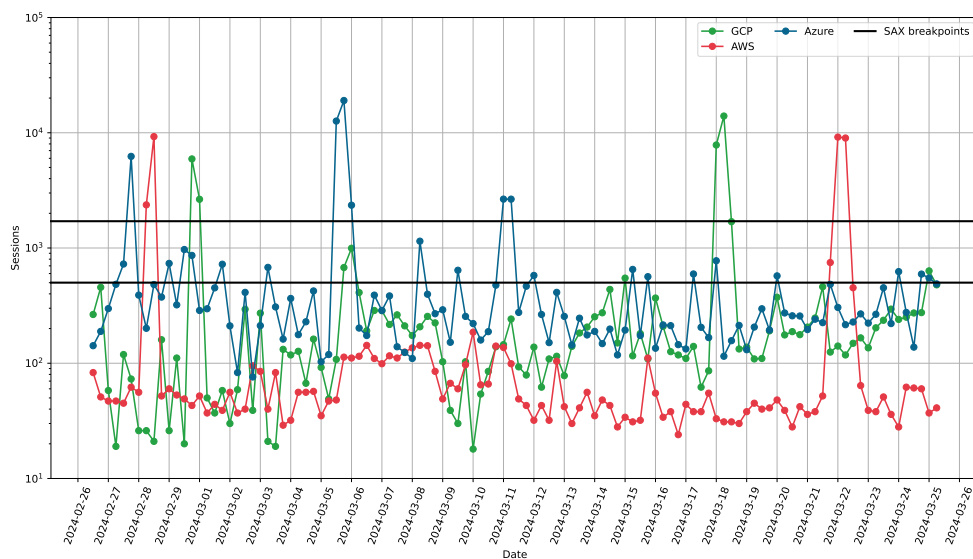


Figure 4.4: **SAX** on **GCP**, **AWS** and **Azure**. The figure describes the number of unique sessions in each 6-hour chunk over the 4-week period for each cloud service provider. The breakpoints for **SAX** are also shown.

Table 4.5: Results from **SAX** for the cloud service providers

	No activity	Continuous activity	Sustained bursts	Ephemeral spikes
GCP	0	103	5	4
AWS	0	107	1	4
Azure	0	89	17	6

4.2 Analysis of Malware

In total, 294 files were downloaded or uploaded across the honeypots. As already stated, all connections to the honeypot are suspect, leading to the belief that all 294 files are malicious. Using cryptographic hash values for the binaries based on their file content, we can confirm that 142 of the 294 files are unique. That is, 152 files are duplicates.

4.2.1 Fingerprinting

The 142 unique files were uploaded to VirusTotal for labelling. Using the label threshold described in 3.4.1, 26 unique cryptojacking malware samples were found. Accounting for duplicate files, a total of 44 cryptojacking malware samples were found. See table 4.7 for a summary of the cryptojacking malware samples. Note that all files are in the **ELF** format, which means all attackers correctly identified the honeypot as a Unix-based system.

See table 4.6 for a summary of the cryptojacking malware on the different cloud service providers, and figure 4.5 for a summary of the origin country of the attacker that downloaded the cryptojacking malware. Interestingly, The Netherlands is highly represented as the origin country of cryptojacking attacks while not appearing as a top attacking country presented in 4.1. Note that only 4 different file names for the cryptojacking malware were found.

Table 4.6: Summary of found cryptojacking malware divided on the cloud service providers

	Total cryptojacking malware	Unique cryptojacking malware	Cryptojacking malware per downloaded file
GCP	11	11	0,11
AWS	14	9	0,23
Azure	19	16	0,15

Table 4.7: Summary of found cryptojacking malware

File name	SHA-256 hash	Malicious labels (Miner)	file type (size)	Platform	Quantity
sshd	94f2e4d8d4436874785cd14e6e6d403507b8750852f7f2040352069a75da4c00	39 / 64 (14)	elf (28.9 mb)	GCP, AWS, Azure	7
redtail.arm8	12ec4ba0eb6c327c01f4b7f9e85a398df8ed1f98e85cfff9f31054e632a35767	33 / 61 (13)	elf (1.52 mb)	Azure	1
redtail.arm7	af985ec54de3697a411f07141bb8771a824929dccc35bda46558dd3f93a01aF9	34 / 63 (12)	elf (1.22 mb)	Azure	1
redtail.i686	4125a1dbd9d9686d68b7b612b661aceeed28b8515b7b46f993b027304c9552b7	35 / 64 (12)	elf (1.77 mb)	GCP, AWS, Azure	4
redtail.i686	adb33719ddca1078bca569e23ac3dc7686886cb0a8e67fe23b0755812b0d0248	34 / 64 (12)	elf (1.79 mb)	Azure	1
sshd	779abd34660f74072b0da4593db8e4639ce1c920ce394ca613dd85f94fc4019b	31 / 64 (12)	elf (18.66 mb)	Azure	1
redtail.x86_64	7636a9c970b9f730fd6d7dbff1c5b9463052528dbfd5813517e7de461c374cfc	30 / 59 (11)	elf (1.81 mb)	Azure	1
redtail.x86_64	6788d005a6ca94eb38038e503f722996744933d60c98539219679b61a563bcf	30 / 64 (11)	elf (1.81 mb)	GCP, AWS, Azure	4
sshd	09d9741f269ede544ba12d8a4478bccda54bb7ae882cd584627ad0bb1b73fd9d4	31 / 64 (11)	elf (4.63 mb)	GCP	1
redtail.x86_64	013f2c8bde97f7d38e31e5f8b17711113bb2f6b1f44a148c520e9a3a2e8ebcd	30 / 63 (11)	elf (1.82 mb)	Azure	1
redtail.arm8	dee9601f6b46b364366544bfbb3f6d9ae5d288c3ade0319b2c6ecfda442519d	29 / 63 (11)	elf (1.51 mb)	Azure	1
sshd	08a899149a76a03dfe3b381e62b45eeaa38c39667a165b3a5a97134349015681	31 / 61 (10)	elf (25.44 mb)	AWS	1
sshd	deaaa6a8a591dc90842c0ed925a3d009eeda0e33e5e71eac91747d6e14465769	27 / 64 (10)	elf (20.44 mb)	AWS	1
sshd	504dce273bdefda628f2d41a5c6d6ec7f48ed44ab17ba666145c322420707210	31 / 63 (10)	elf (19.56 mb)	GCP	1
sshd	a12df8268b2fab5d0c9af3407fd298c862504c4fb75e543c9345185d5d93f2ad	32 / 61 (10)	elf (18.94 mb)	GCP	1
redtail.arm8	3606f13d9b49710da4b67e14d1251872d5119e5094b70ce415de9fe6c6f1e872	30 / 63 (10)	elf (1.52 mb)	GCP, AWS, Azure	4
sshd	3850835d2e9d4a9e710787c0fa048c469cb6fe835e492e550f3356b6ed1dbc3d	29 / 62 (10)	elf (21 mb)	AWS	1
sshd	b6bd684e4d3836547c62aa867903224c0e99a82a7ba0cfff6c7fa247dc094df	27 / 64 (10)	elf (24.19 mb)	AWS	1
redtail.arm7	f8c18b7511e27d5431752c8cb521bea3b0408841f331a89e773918bf5e635cfd	32 / 63 (10)	elf (1.22 mb)	GCP, AWS, Azure	4
sshd	326eea0221b61a452173e9a5e5f515dd42b09f5d5e5f8f868a33e432e4ea18bc3b	30 / 61 (10)	elf (20 mb)	GCP	1
sshd	8f15d36257cc4a9d6e6f94050e6d39193b1968f09c7a9d9d7f83a4556fb4126f82	27 / 64 (10)	elf (11.69 mb)	GCP	1
sshd	dbf08306ed3c5423dadff7e9daa2db0af8a759bc3d085d668a7e57ad4b628f4c	27 / 64 (10)	elf (21.22 mb)	GCP	1
sshd	2cc7249e379420271a3599492b7cfa182251bc66817014699729a2bb346d94ad	31 / 62 (10)	elf (20.19 mb)	Azure	1
redtail.arm7	cfb8a30d6da7903d6775c866723e7a72125343fe7aeb8c113c853b0a2bffc21	28 / 60 (10)	elf (1.22 mb)	Azure	1
redtail.i686	0b9f2aa1b7a475f424787741a5ec58e9cc3ea5d40a42d656ba7252385ee36b4	32 / 62 (9)	elf (1.78 mb)	Azure	1
sshd	815e8321b51a418aeaf5c41c420a595a7ed3e5fb08fd520b55b512e80578ac1d	30 / 62 (9)	elf (26.09 mb)	Azure	1

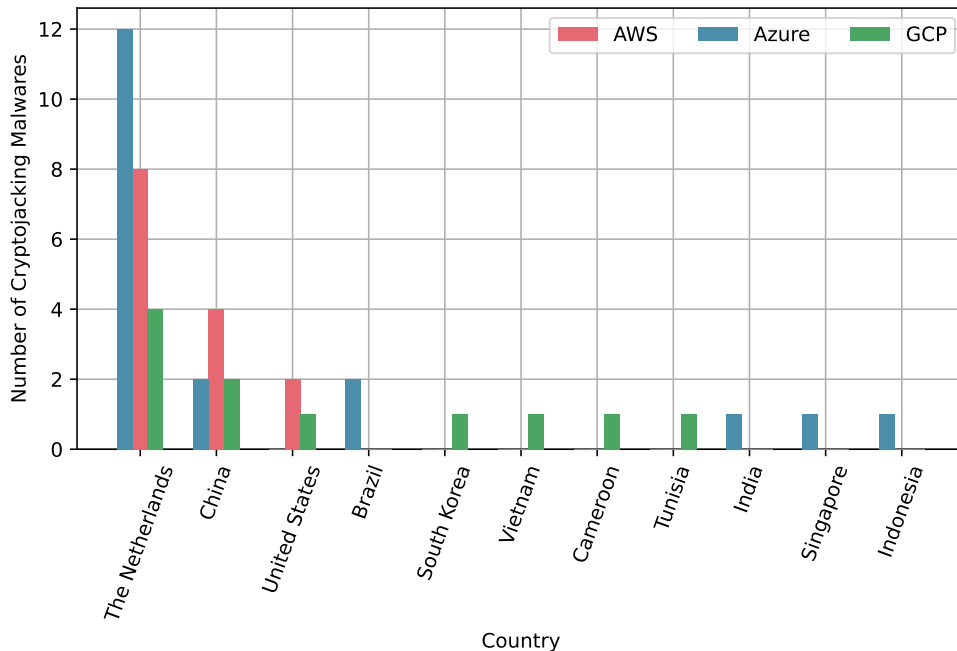


Figure 4.5: Origin countries of cryptojacking attacks

4.2.2 Packer Detection

Out of the 26 unique cryptojacking malware, 12 appear to be packed using **UPX**. Finding a pattern by looking at the file names and sizes is relatively easy. All the file names that include **redtail**, or all the cryptojacking malware that

are less than 2 MB in file size, are packed using **UPX**. The other 14 files appear to be unpacked.

All UPX-packed files managed to be unpacked. Unpacking a file results in a bigger file size and a new SHA-256 checksum. See table 4.8 for the results of the unpacking of the UPX-packed files. Note how the number of miner labels increased drastically while the number of labels overall stayed about the same. **Important:** From now on, the unpacked variant of the UPX-packed files will be referred to as the redtail-files.

Table 4.8: Results of unpacking the **UPX**-packed cryptojacking malware

Old SHA-256 hash	New SHA-256 hash	Malicious labels (Miner)	file type (size)
0b9f2...	6fa07115f8aa0232d4ed804f947d297d726c63fcb7b9fc37793563c08e38bb4	26 / 64 (23)	elf (5.45 mb)
12ec4...	58873a419ddf23fd6e182f3c0e3bc7502652f3c04389e52555c05e3faab3c14b	25 / 62 (21)	elf (4.31 mb)
7636a...	8bbdbd08509003bd273bc4ae906c900076883104d45d359ee3cdf67a1371a4c0	30 / 63 (21)	elf (4.98 mb)
af985...	b2f7e25a60ae0cd672ac21096c410405c22f0e6fb0dd7fba5636710615e2e323	19 / 58 (17)	elf (3.7 mb)
013f2...	91c5d288ba4d4cd76f5017046216925cb5191ed741f2d3ea0b1c8051113a2f3b	31 / 65 (22)	elf (4.97 mb)
3606f...	d3dde936b8cd61c4a1b95e195695439ab2d14e5d44173d7a0e6dc1f1b4fdf6a2	23 / 64 (22)	elf (4.31 mb)
4125a...	45d2d4c3120c5f38179a445db9282ba283c3024090eec72f0939344fa9229790	14 / 52 (12)	elf (5.45 mb)
adb33...	2b2549c34f6534b03250ed18cb75f7217d1856d30e51f9b5b0fc49e1148be1ba	28 / 63 (23)	elf (5.46 mb)
cfb8d...	0055b7f9b57aa6fb057d597276ccdde67010c66d3373a97f716e93f8f55d637d	23 / 63 (19)	elf (3.7 mb)
deef9...	29a388ade836910c3b4ed65e528e503466528d407f1bc46585e312c73a613e76	24 / 62 (21)	elf (4.3 mb)
f8c18...	96e2d1a4b96391a818dd3231d1b7cde6d27df60dfa30511c334f8620720dce9	23 / 64 (19)	elf (3.7 mb)
6788d...	742edeafbfb649b6f614a35ada5860e4e1a072987461285cf74a122ddc73104c	37 / 62 (24)	elf (4.98 mb)

4.2.3 String Extraction

Analysis of the strings extracted from the malware revealed interesting information. All the redtail-files contain the string `XMRIG.VERSION.6.19.2`, suggesting that these files indeed are for cryptojacking purposes and that it is using XMRig 6.19.2 (released 2023-04-03). No other useful information concerning crypto or mining was found within these files.

Extracting strings from the files that were not packed reveals that they all contain the whole or parts of the Base64 encoding of XMRig and NBMiner, whereas NBMiner is a popular **GPU**-based miner. Extracting these Base64 strings, and decoding them gives us the binaries for XMRig 6.16.4 (released 2022-02-04), and NBMiner 40.1 (released 2021-11-24). Some of the files were truncated and did not include all the parts of these base64 encodings. Using a method that involves dividing a file into multiple blocks, with each block having its own hash value calculated, we observe that the initial blocks of these files are identical until some of the files are truncated. Also, the files include other clues about the malware. The string `panchansminingisl` and was found in these files, which could point to them being a part of the Pan-chan botnet. This will further be discussed in section 4.2.6.2. **Important:** From now on, the files without packer will be referred to as the panchan-files.

4.2.4 File Parsing

By extracting metadata from the binaries, it was discovered that the redtail-files are built for the Unix operating system and that they are built for 4 different architectures: ARM, AMD64, i386, and AARCH64. Returning to the Cowrie logs revealed that files for all 4 of the architectures are downloaded simultaneously, along with a setup script that checks the architecture of the current system, executes the correct files and then cleans up all files. The binaries are dynamically linked, meaning the executable relies on external shared libraries, which are loaded into memory at runtime. They are written in C/C++ and are compiled with GCC.

The panchan-files are made for Ubuntu Linux, a Unix-like operating system. They are made for 64-bit systems, and they are statically linked, meaning all necessary library code is included directly within the executable. They are written in Golang, except for the embedded binaries for XMRig and NBMiner, which are written in C/C++ and compiled with GCC.

4.2.5 Disassembly

Disassembling the panchan-files reveals that they are heavily obfuscated, to the point where it is very hard to analyze the disassembled code. However, performing a disassembly on the panchan-files is deemed unnecessary as information could be obtained about the files from security researchers at Akamai, who made a report on the Panchan mining rig, a peer-to-peer botnet and SSH worm [58].

Disassembling the redtail-files reveals that they also are heavily obfuscated, and extracting something useful would be very challenging. Traces of base64 strings could be found around the binaries but were unable to be reconstructed by following the flow of the disassembled code. The control-flow graphs of the redtail-files suggest that the files that are made for the same architecture share identical program logic and structure. In other words, it is probable that they are the same malware but obfuscated to new versions. See Appendix B for the control flow graphs.

4.2.6 Malware Analysis Summary

The malware analysis suggests that we only deal with 2 different types of cryptojacking malware families, Redtail and Panchan. Note that these are not widely established names for the families, as little is known about them, but the names will continue to be used in this thesis.

4.2.6.1 Redtail

In total, 24 redtail-files were observed, where 12 were unique files. They are **ELF** files, written in C/C++, packed with UPX and built for 4 different architectures. They contain the XMRig 6.19.2 and are heavily obfuscated to the point where the disassembled code is hard to analyze. All 24 redtail-files originated from the same IP address from The Netherlands.

The binaries for all 4 architectures are downloaded simultaneously, together with a setup script and an SSH-key. That means 3 unique sets of redtail-files were obtained, and they show indications that led us to believe that the sets are obfuscated versions of each other.

The setup script shows us that the binaries should be executed with *ssh* as the one and only argument. Additionally, The name of the SSH-key includes 20230629, which could be a clue that these redtail-files started to spread 2023-06-29.

From the information obtained, one blog post could be found covering the Redtail cryptominers [59]. From the blog post, not much information about the malware is obtained, other than that the attackers were using the CVE-2022-23329 exploit, which allowed attackers to execute arbitrary commands via uploading malicious files.

4.2.6.2 Panchan

In total, 20 panchan-files were obtained, where 13 were unique files. They are **ELF** files, written in Golang and built for AMD64. They contain the XMRig 6.16.4 and NBMiner 40.1, encoded as base64 strings. Many Panchan samples are likely truncated versions of a complete file caused by the attacker timing out from the server while uploading the file. The 20 panchan-files originated from 18 unique IP addresses and 10 different countries.

Returning to the Cowrie logs reveals that the binaries are attempted to be executed with 51 IP addresses as arguments, likely peers in the botnet.

The Akamai report [58] suggests that the Panchan campaign started in March 2022, and except for cryptojacking, the malware is designed to perform lateral movement to expand the mining. During runtime, the miners are not extracted to the disk at all, instead they are memory-mapped so it can be executed directly from memory without having a traceable file path.

4.3 Mitigation of Cryptojacking Attacks

To detect and mitigate these cryptojacking families, Redtail and Panchan, **YARA** rules were developed. First, the **YARA** rule to detect Panchan samples found in the Akamai report [58], and the **YARA** rule to detect Redtail samples found in the Exylum blog post [59], were confirmed not to detect our samples from the Panchan and Redtail family. No other **YARA** rules could be found to detect these cryptojacking families.

4.3.1 Development of YARA Rules

The constructed rules are the results of modifying the existing **YARA** rules for each family and incorporating the findings from the malware analysis in this study. The result is one rule for each family. See listing 4.2 and listing 4.3. Note that **IoCs** longer than 25 bytes have been truncated. The rules in their entirety can be found in the GitHub repository in Appendix A.

The first four bytes of an **ELF** file are what's referred to as the **ELF** magic number: 7F 45 4C 46. These bytes identify the file as in the **ELF** format.

The Redtail rule comprises the **ELF** magic and the ASCII string `xmrig` (case-insensitive). Thereafter, it must contain one of the entries defined for the 4 different architectures. A pattern could be found in the obfuscations of the binaries where bytes with the same offsets from the entry point were tampered with. Therefore, some bytes are replaced with `?` in the rules. The entry point for each binary was different, but the `elf` module in **YARA** can proceed from the actual entry point of the binary.

The Panchan rule also consists of the **ELF** magic and the ASCII string `xmrig` (case-insensitive), but also contains the ASCII string `nbminer` (case-insensitive). Furthermore, the characteristic of being a Go binary was incorporated, as `goroutine` is directly integrated into the Go runtime, which is included in all Go binaries. An identified string in all Panchan binaries was the `"pan-chan's mining _ hi!"`, where `_` would either be `"rig"` or `"island"`. Lastly, since the `XMRig` and `NBMiner` were embedded as base64 strings, the base64 table header is included.

Listing 4.2: YARA rule to detect Redtail cryptojacking malware

```

rule Detect_Redtail_Malware {
  meta:
    author = "Edvin Baggman"
    filetype = "ELF"
    date = "2024-05-06"
    malware = "REDTAIL"
    description = "YARA to detect Redtail
                  cryptojacking malware"

  strings:
    $self_magic = { 7f 45 4c 46 }
    $str_xmrig = "xmrig" nocase
    $entry_386 = { 31 ED 89 E0 83 E4 F0 50 50
                  E8 00 00 00 00 81 04 24 ?? ?? ?? 00 50
                  E8 00 00 ... }
    $entry_AMD64 = { 48 31 ED 48 89 E7 48 8D 35
                    ?? ?? ?? 00 48 83 E4 F0 E8 00 00 00 00
                    48 81 EC ... }
    $entry_ARM = { 00 B0 A0 E3 00 E0 A0 E3 10
                  10 9F E5 01 10 8F E0 0D 00 A0 E1 0F C0
                  C0 E3 0C ... }
    $entry_AARCH64 = { 1D 00 80 D2 1E 00 80 D2
                      E0 03 00 91 ?1 21 00 ?0 21 ?0 2? 91 1F
                      EC 7C 92 0B ... }

  condition:
    $self_magic at 0 and
    $str_xmrig and
    any of ($entry_* ) at elf.entry_point
}

```

Listing 4.3: YARA rule to detect Panchan cryptojacking malware

```

rule Detect_Panchan_Malware {
  meta:
    author = "Edvin Baggman"
    filetype = "ELF"
    date = "2024-05-06"
    malware = "PANCHAN"
    description = "YARA to detect Panchan
                  cryptojacking malware"

  strings:
    $self_magic = { 7f 45 4c 46 }
    $str_nbminer = "nbminer" nocase
    $str_xmrig = "xmrig" nocase
    $golang = "goroutine"
    $hex_panchan_header = { 70 61 6E 2D 63 68
                           61 6E 27 73 20 6D 69 6E 69 6E 67 20
                           [0-10] 20 68 69 21} // pan-chan's mining
                           _ hi!
    $base64_table = { 41 42 43 44 45 46 47 48
                     49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55
                     56 57 58 59 ... }

  condition:
    all of them and $self_magic at 0
}

```

4.3.2 Evaluation of YARA Rules

To evaluate the rules, a controlled dataset of malware samples needs to be constructed that does not include any of the samples used to create the rules. The malware database MalwareBazaar [57] was used.

Having identified 2 cryptojacking families, the categories used for the dataset were: *Redtail*, *Panchan*, *Other*, *Unknown*. The category *Other* holds all samples that can be confirmed to belong to another cryptojacking family other than Redtail or Panchan. The category *Unknown* holds all samples that cannot be confirmed to be in one of the families nor be confirmed to be outside the families. These samples will not be used when evaluating the rules.

First, a search for the tag `coinminer` was made, and all `.elf` files in the period 2023-02-20 - 2024-05-09 was downloaded. A total of 74 files were downloaded. 3 samples were duplicates from this study and will therefore not be used. Second, a search for the tag `panchan` was made, and all 49 samples were downloaded. The period in which these files were uploaded were 2022-07-16 - 2024-04-12.

Through analysis, as described in Section 3.5.2, 50 samples were placed in the *Panchan* category, 38 samples in the *Redtail* category, 25 samples in the *Other* category, and 7 samples in the *Unknown* category. In total, 113 samples will be used when evaluating the rules.

Applying the **YARA** rules to this dataset provided the results presented in table 4.9. Additionally, the **YARA** rules found in the Akamai report [58] and the Exylum blog post [59] were applied to the dataset. Evaluation metrics were calculated and are presented in table 4.10. The rules constructed in this study demonstrate significantly improved performance on these from-the-wild samples.

Table 4.9: Detection results of YARA rules

	Total samples	Panchan rule	Redtail rule	Akamai Panchan rule	Exylum Redtail rule
Panchan	50	50	0	2	0
Redtail	38	0	35	0	4
Other	25	0	0	0	0

Table 4.10: Evaluation metrics of YARA rules

	Panchan rule	Redtail rule	Akamai Panchan rule	Exylum Redtail rule
Accuracy	100%	97.4%	57.5%	69.9%
Precision	100%	100%	100%	100%
Recall	100%	92.1%	4%	10.5%
F1-score	100%	95.9%	7.7%	19.1%

4.4 Validity and Reliability Analysis

Out of the 142 unique files in the honeypots, only 26 were fingerprinted as cryptojacking malware. This number might be underestimated as some of

the files found in the honeypots were scripts to download files, potentially cryptojacking malware. However, these scripts were not analyzed further.

The results of applying the **YARA** rules on this dataset might not accurately reflect their real performance. Due to time constraints, only 113 samples were used in the tests. The process of classifying malware manually proved very time-consuming, and it's not guaranteed more samples would be found if multiple malware databases were used. Ideally, the dataset would be many times larger and include other types of malware and benign files to match the wide variety of malware found in real-world scenarios.

Both malware disassembly and **YARA** rule development are highly specialized skills that require significant time and expertise to master. Given the frame of this project, it would be unrealistic to achieve a comprehensive malware disassembly and to translate that into more effective **YARA** rules.

Chapter 5

Discussion

In this chapter, a discussion of the results relating to the method and literature is presented.

5.1 Overall Results

In total, the honeypots observed approximately 170,000 successful access attempts solely due to the configuration of weak credentials. This significantly exceeds initial expectations one might have when deploying a server on a cloud service provider. The IP address of the system does not need to be published anywhere; adversaries will find it anyway.

This observation highlights the ongoing importance of cybersecurity in our society. Having access to a system, a cryptojacking attack is just one out of many attacks that an attacker can perform. It is noteworthy that cryptojacking attacks were nonexistent a decade ago. Looking ahead, it is difficult to predict the type of threats that may appear.

5.2 Attacks on Cloud Service Providers

The analysis of the honeypot data indicates discernible differences in the threat landscape between the evaluated cloud service providers **GCP**, **AWS** and **Azure**.

Examining the sources of attacks, the top attacking countries are not consistent across providers, as **AWS** observed more attacks from India than South Korea, contrary to **GCP** and **Azure**. While this discrepancy might be coincidental, it is intriguing that the origin of attacks varies.

In terms of attack frequency, more than double the activity could be found in the **Azure** honeypot than in the **AWS** honeypot, with **GCP** falling between the two. When analyzing attack patterns, it is evident that **Azure** experiences significantly more sustained bursts of attacks than **GCP** and **AWS**. Still, the ephemeral spikes are consistent between the providers. The underlying reasons for these variations are not immediately clear. However, the increased attacks on **Azure** don't necessarily indicate that it is less secure; it is just an indication of the current threat landscape.

Interestingly, no correlation could be identified among the providers, indicating that the attacks on them are independent of one another. Furthermore, the low proportion overlap of attacking IP addresses on the cloud service providers supports this theory of attacks on different cloud service providers being independent. This implies that the occurrence of an attack on one provider does not influence nor predict the likelihood of an attack on another provider.

The propagation graphs indicate that attackers employ different tactics towards the various providers. Specifically, Attackers seem more likely to use a new IP address when re-attacking the same honeypot on **AWS** than on **GCP** and **Azure**. This raises the question if attackers on **AWS** are more sophisticated, utilizing new IP addresses, or if they abandon their efforts more quickly. This variation in behaviour suggests varying strategies among the attackers.

The findings of this study present a completely different picture compared to Kelly et al. in their comparison of different cloud platforms from 2021 [21], which reported that **GCP** experienced 10 to 20 times more attacks than **AWS** and **Azure**. Additionally, this study retrieved more than double the number of suspicious files in about half the amount of time compared to the Baser et al. study from 2023, which focuses on analysing malicious files gathered via a honeypot on **GCP** [43]. It is difficult to determine whether these differences are attributable to the continuously evolving threat landscape or to the efforts in configuring and obscuring the honeypots in this study. Regardless, these results should provide interesting information about the current threat landscape of cloud service providers.

5.3 Cryptojacking on Cloud Systems

During the 4-week period, two cryptojacking campaigns targeting Unix-based systems were believed to have been detected. Indications suggest that the Redtail campaign commenced in June 2023, and the Panchan campaign began

in March 2022.

All honeypots experienced attacks involving cryptojacking malware. Both cryptojacking families, Panchan and Redtail, were observed across all cloud service providers. 20 Panchan attacks were recorded, while just 6 Redtail attacks were recorded. The two cryptojacking families employ different attack strategies as all 6 Redtail attacks originate from the same IP address, and the 20 Panchan attacks originate from 18 unique IP addresses. Attacks from both families accessed the systems through poorly configured credentials. The reports covering the cryptojacking families [58, 59] suggest that the attacks also employ other strategies, including a novel credential harvesting method and exploitation of CVE-2022-23329, which allowed attackers to execute arbitrary commands via uploading malicious files.

This study does not address the prevalence of cryptojacking attacks towards systems other than systems with Unix-based OSes. Browsing malware databases such as MalwareBazaar [57] reveals that most uploaded cryptojacking malware is in the .exe format, thus targeting Windows systems. The study by Jayasinghe and Poravi, documenting attack instances of cryptojacking targeting cloud infrastructure in 2020 [38], also observed OSes differences, where most attacks targeted Windows. The reason for this appears unclear, as the background suggests Linux as the most widely used operating system for systems in the cloud. It raises the question of what amount and what families of cryptojacking attacks might have been observed, given that Windows-based honeypots were deployed.

A common command executed by attackers was `uname`, which provides information about the system. That raises the question of how the configuration of the system influences the attacks. Could a more powerful system appeal to other families of cryptojacking that were not detected in this study? Additionally, the study did not analyze the download scripts found in the honeypots, which could have uncovered more cryptojacking attacks and possibly identified additional cryptojacking families.

Similar to the study by Zimba et al. investigating state-of-the-art crypto mining attacks in 2018 [52] and the study by Jayasinghe and Poravi documenting attack instances of cryptojacking targeting cloud infrastructure in 2020 [38], the software XMRig is observed in every cryptojacking sample. Intriguingly, XMRig persists as the default choice for cryptojacking attacks. However, NBMiner was not mentioned in their investigations as a utility used in cryptojacking attacks. Assessing the prevalence of NBMiner in state-of-the-art cryptojacking attacks today is difficult, as only 2 different families are analyzed in this study. Still, it is clear that NBMiner is indeed utilized within

cryptojacking attacks.

In summary, the three Unix-based cloud systems with poorly configured credentials experienced at least 26 cryptojacking attacks during the 4-week period, corresponding to an average of around one cryptojacking attack every three days per system.

5.4 Mitigation of Cryptojacking Attacks

Following the analysis of the two cryptojacking families, **YARA** rules were developed to detect and mitigate these families. The results indicate that these rules perform significantly better than those available on the web. This suggests that data collected by honeypots can indeed aid in improving current cryptojacking detection solutions, particularly by improving **YARA** rules.

The developed rules are not perfect and do not necessarily contain the most optimal **IoCs**. They are the outcome of a study covering cryptojacking in cloud systems. Developing effective **YARA** rules requires specialized skills that take significant time to master, and expecting precise rules would be unrealistic. For instance, it might not be advisable to use lengthy **IoCs** as in the case of the entries of the redtail-files. Longer **IoCs** can be more specific, which might seem beneficial at first glance. However, this specificity can lead to increased false negatives, where a variant of the malware is missed because it doesn't match the exact long **IoC**. An alternative approach would be to identify other **IoCs** common in all redtail-files, regardless of the architecture.

The F1-score of the rules developed in this study indicates that they perform well on the dataset. The increase from 7.69% to 100% for the Akamai Panchan rule and from 19.05% to 95.89% for the Exylum Redtail rule demonstrates a drastic increase in effectiveness. However, a reasonable area of concern is the quality of the dataset and its resemblance to a real-world scenario. As discussed in section 4.4, best efforts were made in creating the dataset, but due to time limitations and a shortage of cryptojacking samples available, the dataset is relatively small.

It's also important to acknowledge that these results do not guarantee the future performance of the rules, as the cryptojacking families can continue to evolve into variants not recognised by the **YARA** rules. For instance, the disassembly of the panchan-files showed that heavy obfuscations had been performed. It is unreasonable to believe that a **YARA** rule can detect 100% of a malware family that gets obfuscated before new attacks. Therefore, these results should be interpreted with caution and viewed more as a conceptual demonstration of how honeypot data can aid in the goal of improving existing

malware detection solutions as threats evolve.

To the author's best knowledge, there does not seem to exist better **YARA** rules to identify Panchan and Redtail samples. Still, people may have developed their own rules for this purpose without making them public, as that information could be used by the attackers themselves to mitigate detection. That also means that the rules developed in this study might be accessed by the attackers, who may modify the malware accordingly to circumvent detection.

Chapter 6

Conclusions and Future work

This final chapter of the thesis consists of conclusions, limitations of the study, future work, and final reflections.

6.1 Conclusions

The problem the thesis addresses is the limited research on the prevalence of cryptojacking attacks on cloud systems. 44 Cryptojacking malware samples caught in honeypots deployed on cloud service providers were analyzed to improve a cryptojacking mitigation technique and tested on a dataset of 113 cryptojacking samples. The thesis posed three research questions with the following conclusions:

RQ 1 *What are the differences between the threat landscapes found by honeypots across different cloud service providers?*

Distinct differences in the threat landscapes across various cloud service providers could be found by honeypots, with significant variations in attack frequencies and attack patterns. Additionally, the study suggests that the attacks on the different cloud service providers are mostly independent of one another. In the study, the cloud service providers **Azure**, **GCP** and **AWS** were used, experiencing number of attacks in that order, from most to least.

RQ 2 *What are the prevalence and patterns of cryptojacking attacks on cloud systems?*

The study demonstrated a notable prevalence of cryptojacking attacks on cloud systems during the period the study was made, capturing a total of 26

cryptojacking attacks on three honeypots over a 4-week period. That averages out to around **one cryptojacking attack every three days** per system. The analysis revealed that these samples belonged to two primary cryptojacking families, Panchan and Redtail, using the crypto mining utilities XMRig and NBMiner, indicating distinct attack patterns.

RQ 3 *What is a suitable method for utilizing honeypot data to improve security measures against cryptojacking?*

The study shows that developing **YARA** rules is a suitable and effective method for utilizing honeypot data to improve security measures against cryptojacking. By analyzing cryptojacking malware caught in the honeypots, existing **YARA** rules could be drastically improved to detect the cryptojacking families observed.

6.2 Limitations

This study has two limitations that directly affect the results of the study.

First, as stated in Section 3.2, the honeypot utilized in this study emulates a Unix-based system only. Consequently, it is inherently designed to attract attacks targeting that particular system. This limitation significantly restricts the generalizability of the findings to all cloud systems, as not all cloud environments operate on Unix-based **OSes**. Therefore, the conclusions drawn in response to Research Question 2 cannot be said to address all cloud systems.

Additionally, as stated in Section 3.2.2, it is important to acknowledge the uncertainty regarding the number of attacks that may have been aborted due to honeypot fingerprinting. It is theoretically possible that numerous active cryptojacking campaigns exist, operating with a greater ability to identify whether a targeted system is indeed a honeypot. This uncertainty complicates the generalization of the findings in response to Research Question 2.

6.3 Future work

The threat landscape is ever-changing, and staying up to date is essential for addressing current challenges. Several possible continuations can build upon the work presented in the thesis. These extensions address the limitations discussed earlier or address the work in a different context.

Specifically, one could investigate various configurations of honeypots. The frequency of `uname` commands observed suggests that the configuration

of the honeypots could significantly impact the attacks observed. A relevant research question could be formulated as follows: *How do the configurations of honeypots influence the scope of attacks observed?*. This could be investigated by comparing different types of CPUs configured on the honeypots or comparing different OSes by also incorporating a honeypot emulating a Windows system to contrast with Unix-based systems.

Or, one could investigate the difference in attacks observed using a honeypot and a genuine vulnerable system. As stated in Section 3.2.2, this would demand considerable time and resources, but it holds the potential to either affirm or challenge findings such as those presented in this study.

Another continuation of this work could involve enhancing the YARA rules and, most importantly, expanding the dataset on which the rules are tested. Additionally, it would be valuable to explore alternative methods through which the results from the malware analysis can aid in mitigating cryptojacking malware. While YARA is a tool where researchers can quickly contribute with rules, it is likely that there are other ways where security researchers can readily contribute with their findings.

Finally, it could be interesting to assess the effectiveness of current state-of-the-art cryptojacking detectors with cryptojacking samples caught in the honeypots. However, the pre-study during this project revealed that it's challenging to find host-based cryptojacking detectors available for public use. Nonetheless, if suitable detectors can be identified, evaluating their performance with real-world cryptojacking samples could provide valuable insights into their efficacy and potential areas for improvement.

6.4 Reflections

This thesis presents two significant contributions to the field of security. First, it contributes to the ability to identify the cryptojacking families Panchan and Redtail. Second, it provides updated information about the threat landscape on cloud service providers. This research is important not only to secure people from miners on their servers but also because crypto mining is a large environmental problem in terms of energy consumption.

The insights obtained from the study hold interest for the general public, as they shed light on the significant volume of cyber-attacks executed daily. Increased awareness and carefulness among individuals would help protect the general public against unethical hackers.

The long-term relevance of the study depends on the continued relevance and value of crypto mining in the future. The value of cryptocurrencies

fluctuates greatly, and there are no guarantees that they will hold their value. Nonetheless, the study provides a concept of how honeypot data can be analysed to improve current malware detection solutions.

References

- [1] A. Sunyaev, “Cloud computing,” in *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, Cham: Springer International Publishing, 2020, pp. 195–236, ISBN: 978-3-030-34957-8. DOI: [10.1007/978-3-030-34957-8_7](https://doi.org/10.1007/978-3-030-34957-8_7). [Online]. Available: https://doi.org/10.1007/978-3-030-34957-8_7.
- [2] H. Tabrizchi and M. Kuchaki Rafsanjani, “A survey on security challenges in cloud computing: Issues, threats, and solutions,” *The journal of supercomputing*, vol. 76, no. 12, pp. 9493–9532, 2020, Publisher: Springer.
- [3] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, “SoK: Cryptojacking malware,” in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2021, pp. 120–139.
- [4] Check Point Research, “2024 cyber security report,” Check Point Software Technologies Ltd., 2024. [Online]. Available: <https://pages.checkpoint.com/2024-cyber-security-report>.
- [5] Check Point Research, “2023 cyber security report,” Check Point Software Technologies Ltd., 2023. [Online]. Available: <https://resources.checkpoint.com/report/2023-check-point-cyber-security-report>.
- [6] Check Point Research, “2022 cyber security report,” Check Point Software Technologies Ltd., 2022. [Online]. Available: <https://resources.checkpoint.com/cyber-security-resources/2022-cyber-security-report>.
- [7] H. Almurshi, “Smart analysis and detection system for new host-based cryptojacking malware dataset,” *Journal of Engineering and Applied Sciences*, vol. 10, no. 1, p. 69, 2023.

- [8] Ö. A. Aslan and R. Samet, “A comprehensive review on malware detection approaches,” *IEEE access*, vol. 8, pp. 6249–6271, 2020, Publisher: IEEE.
- [9] J. Franco, A. Acar, A. Aris, and S. Uluagac, “Forensic analysis of cryptojacking in host-based docker containers using honeypots,” in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 4860–4865. DOI: [10.1109/ICC45041.2023.10278764](https://doi.org/10.1109/ICC45041.2023.10278764).
- [10] M. Oosterhof and C. contributors, *Cowrie honeypot*, 2024. [Online]. Available: <https://github.com/cowrie/cowrie> (visited on 02/21/2024).
- [11] M. Oosterhof, *Cowrie documentation*, 2024. [Online]. Available: <https://readthedocs.org/projects/cowrie/downloads/pdf/latest> (visited on 02/21/2024).
- [12] D. Ucuz *et al.*, “Comparison of the IoT platform vendors, microsoft azure, amazon web services, and google cloud, from users’ perspectives,” in *2020 8th international symposium on digital forensics and security (ISDFS)*, IEEE, 2020, pp. 1–4.
- [13] D. Bove and T. Müller, “Investigating characteristics of attacks on public cloud systems,” in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2019, pp. 89–94. DOI: [10.1109/CSCloud/EdgeCom.2019.00-13](https://doi.org/10.1109/CSCloud/EdgeCom.2019.00-13).
- [14] A. Wittig and M. Wittig, *Amazon Web Services in Action: An in-depth guide to AWS*. Simon and Schuster, 2023.
- [15] S. Kaur and G. Kaur, “Threat and vulnerability analysis of cloud platform: A user perspective,” in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 533–539.
- [16] J. Agorye. “Operating systems common in cloud data centers,” Verpex. (May 23, 2023), [Online]. Available: <https://verpex.com/blog/cloud-hosting/operating-systems-common-in-cloud-data-centers> (visited on 05/21/2024).
- [17] H. M. Almohri, L. T. Watson, and D. Evans, “Predictability of IP address allocations for cloud computing platforms,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 500–511, 2019, Publisher: IEEE.

- [18] F. K. M. Zainudin, I. H. Ishak, S. Sulaman, F. Ramlee, N. S. Jamaludin, and S. Chantando, "Malware discovery using lebahnet technology," *OIC-CERT Journal of Cyber Security*, vol. 2, no. 1, pp. 69–76, 2020.
- [19] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.
- [20] K. Jiang and H. Zheng, "Design and implementation of a machine learning enhanced web honeypot system," in *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2020, pp. 957–961. doi: [10.1109/CISP-BMEI51763.2020.9263640](https://doi.org/10.1109/CISP-BMEI51763.2020.9263640).
- [21] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, "A comparative analysis of honeypots on different cloud platforms," *Sensors*, vol. 21, no. 7, p. 2433, 2021, Publisher: MDPI.
- [22] S. Machmeier, "Honeypot implementation in a cloud environment," *arXiv preprint arXiv:2301.00710*, 2023.
- [23] W. Cabral, C. Valli, L. Sikos, and S. Wakeling, "Review and analysis of cowrie artefacts and their potential to be used deceptively," in *2019 International Conference on computational science and computational intelligence (CSCI)*, IEEE, 2019, pp. 166–171.
- [24] T. Sochor and M. Zuzcak, "Study of internet threats and attack methods using honeypots and honeynets," in *Computer Networks*, A. Kwiecień, P. Gaj, and P. Stera, Eds., Cham: Springer International Publishing, 2014, pp. 118–127, ISBN: 978-3-319-07941-7.
- [25] H. Arif, F. Hoda, and A. Kumar, "Establishing cloud security by setting up honeypot on azure services," 2023.
- [26] P. Sokol, J. Míšek, and M. Husák, "Honeypots and honeynets: Issues of privacy," *EURASIP Journal on Information Security*, vol. 2017, pp. 1–9, 2017, Publisher: Springer.
- [27] M. Sladić, V. Valeros, C. Catania, and S. Garcia, "LLM in the shell: Generative honeypots," *arXiv preprint arXiv:2309.00155*, 2023.
- [28] S. C. Sethuraman, T. G. Jadapalli, D. P. V. Sudhakaran, and S. P. Mohanty, "Flow based containerized honeypot approach for network traffic analysis: An empirical study," *Computer Science Review*, vol. 50, p. 100600, 2023, Publisher: Elsevier.

- [29] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, “Gotta catch ’em all: A multistage framework for honeypot fingerprinting,” *Digital Threats*, vol. 4, no. 3, Oct. 2023, Place: New York, NY, USA Publisher: Association for Computing Machinery. doi: 10.1145/3584976. [Online]. Available: <https://doi.org/10.1145/3584976>.
- [30] W. Z. Cabral, C. Valli, L. F. Sikos, and S. G. Wakeling, “Advanced cowrie configuration to increase honeypot deceptiveness,” in *ICT Systems Security and Privacy Protection*, A. Jøsang, L. Fitcher, and J. Hagen, Eds., Cham: Springer International Publishing, 2021, pp. 317–331, ISBN: 978-3-030-78120-0.
- [31] boscutti939 and 411Hall, *Cowrie obscurer*, 2020. [Online]. Available: <https://github.com/boscutti939/obscurer> (visited on 02/21/2024).
- [32] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, “A brief survey of cryptocurrency systems,” in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016, pp. 745–752. doi: 10.1109/PST.2016.7906988.
- [33] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*, 2008.
- [34] J. Wu, J. Liu, Y. Zhao, and Z. Zheng, “Analysis of cryptocurrency transactions from a network perspective: An overview,” *Journal of Network and Computer Applications*, vol. 190, p. 103 139, 2021, ISSN: 1084-8045. doi: <https://doi.org/10.1016/j.jnca.2021.103139>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804521001557>.
- [35] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.
- [36] S.-N. Li, Z. Yang, and C. J. Tessone, “Proof-of-work cryptocurrency mining: A statistical approach to fairness,” in *2020 IEEE/CIC international conference on communications in China (ICCC workshops)*, IEEE, 2020, pp. 156–161.
- [37] R. Tahir *et al.*, “Mining on someone else’s dime: Mitigating covert mining operations in clouds and enterprises,” in *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, 2017, pp. 287–310.

- [38] K. Jayasinghe and G. Poravi, “A survey of attack instances of cryptojacking targeting cloud infrastructure,” in *Proceedings of the 2020 2nd Asia pacific information technology conference*, 2020, pp. 100–107.
- [39] T. Muralidharan, A. Cohen, N. Gerson, and N. Nissim, “File packing from the malware perspective: Techniques, analysis approaches, and directions for enhancements,” *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–45, 2022, Publisher: ACM New York, NY.
- [40] P. Patel, A. Dalvi, and I. Siddavatam, “Exploiting honeypot for cryptojacking: The other side of the story of honeypot deployment,” in *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2022, pp. 1–5. doi: [10.1109/ICCUBEA54992.2022.10010904](https://doi.org/10.1109/ICCUBEA54992.2022.10010904).
- [41] A. Lockett, “Assessing the effectiveness of yara rules for signature-based malware detection and classification,” *arXiv preprint arXiv:2111.13910*, 2021.
- [42] H. Darabian *et al.*, “Detecting cryptomining malware: A deep learning approach for static and dynamic analysis,” *Journal of Grid Computing*, vol. 18, pp. 293–303, 2020, Publisher: Springer.
- [43] M. BASER, E. Y. GUVEN, and M. A. AYDIN, “Analysis of malicious files gathering via honeypot trap system and benchmark of anti-virus software,” 2023.
- [44] I. M. M. Matin and B. Rahardjo, “Malware detection using honeypot and machine learning,” in *2019 7th international conference on cyber and IT service management (CITSM)*, vol. 7, IEEE, 2019, pp. 1–4.
- [45] VirusTotal. “Online virus scan and analysis service.” (2024), [Online]. Available: <https://www.virustotal.com> (visited on 03/06/2024).
- [46] P. Peng, L. Yang, L. Song, and G. Wang, “Opening the blackbox of virustotal: Analyzing online phishing scan engines,” in *Proceedings of the Internet Measurement Conference*, 2019, pp. 478–485.
- [47] J. Cabrera-Arteaga, M. Monperrus, T. Toady, and B. Baudry, “WebAssembly diversification for malware evasion,” *Computers & Security*, vol. 131, p. 103 296, 2023, issn: 0167-4048. doi: <https://doi.org/10.1016/j.cose.2023.103296>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404823002067>.

- [48] N. Naik *et al.*, “Embedded YARA rules: Strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis,” *Complex & Intelligent Systems*, vol. 7, pp. 687–702, 2021, Publisher: Springer.
- [49] E. Raff *et al.*, “Automatic yara rule generation using biclustering,” in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, 2020, pp. 71–82.
- [50] N. Naik, P. Jenkins, R. Cooke, J. Gillett, and Y. Jin, “Evaluating automatically generated YARA rules and enhancing their effectiveness,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2020, pp. 1146–1153.
- [51] Deutsche Telekom Security GmbH and M. Ochse, *T-pot*, version 22.04.0, Apr. 2022. [Online]. Available: <https://github.com/telekom-security/tpotce>.
- [52] A. Zimba, Z. Wang, M. Mulenga, and N. H. Odongo, “Crypto mining attacks in information systems: An emerging threat to cyber security,” *Journal of Computer Information Systems*, 2018, Publisher: Taylor & Francis.
- [53] O. Thonnard and M. Dacier, “A framework for attack patterns’ discovery in honeynet data. DFRWS 2008,” in *8th Digital Forensics Research Conference*, vol. 113, 2008, p. 114.
- [54] horsicq, *Detect-it-easy*, 2024. [Online]. Available: <https://github.com/horsicq/Detect-It-Easy>.
- [55] National Security Agency, *Ghidra*, Mar. 2019. [Online]. Available: <https://github.com/NationalSecurityAgency/ghidra> (visited on 04/22/2024).
- [56] S. Zhu *et al.*, “Measuring and modeling the label dynamics of online {anti-malware} engines,” in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2361–2378.
- [57] Abuse.ch. “Malwarebazaar.” Accessed: 2024-05-10. (2024), [Online]. Available: <https://bazaar.abuse.ch>.
- [58] S. Kupchik. “Panchan’s mining rig: New golang peer-to-peer botnet says “hi!”” Akamai. (Jun. 15, 2022), [Online]. Available: <https://www.akamai.com/blog/security-research/new-p2p-botnet-panchan> (visited on 04/30/2024).

- [59] “Interesting URL strings used to download redbtail cryptominers,” Exylum Technical. (Dec. 26, 2023), [Online]. Available: <https://exylum.tech/blog/scanning-23-12.html> (visited on 05/06/2024).

Appendix A

Supporting materials

YARA rules and code for the scripts used for honeypot log analysis can be found at:


<https://github.com/edvinbaggman/exjobb>

Forked Cowrie Obscurer repository can be found at:

<https://github.com/edvinbaggman/cowrie-obscurer>

Forked Cowrie logviewer used to visualize logs repository can be found at:

<https://github.com/edvinbaggman/cowrie-logviewer>

The BibTeX references used in this thesis are attached. 

Appendix B

Control-flow graph of entry point for redtail-files

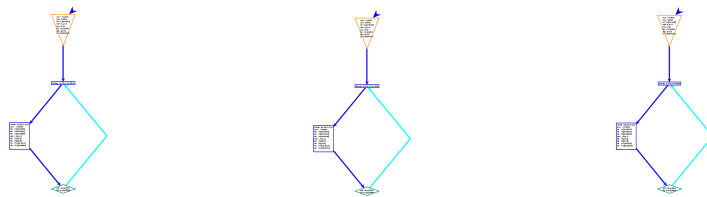


Figure B.1: Control-flow graph of entry point for redtail-files for ARM

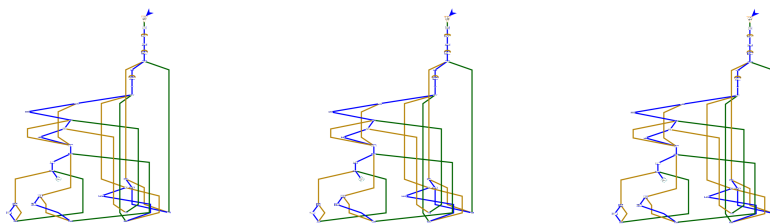


Figure B.2: Control-flow graph of entry point for redtail-files for AMD64

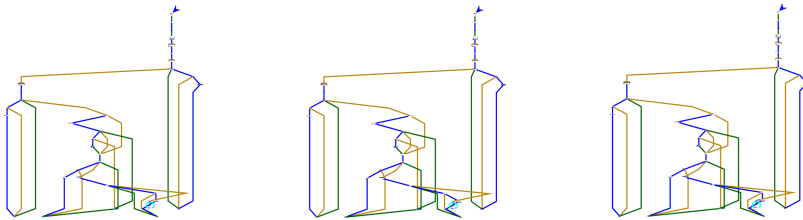


Figure B.3: Control-flow graph of entry point for redtail-files for i386

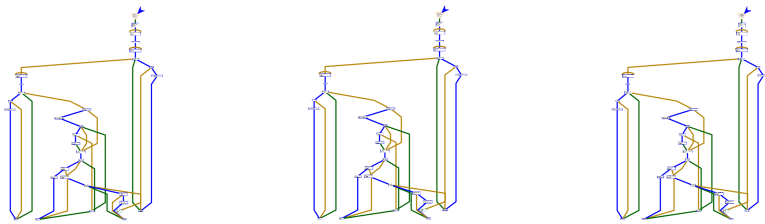


Figure B.4: Control-flow graph of entry point for redtail-files for AARCH64

TRITA-EECS-EX-2024:478
Stockholm, Sweden 2024

www.kth.se