11-3-2021

A Deep-dive into Cryptojacking Malware: From an Empirical Analysis to a Detection Method for Computationally Weak Devices

Ege Tekiner
Florida International University, eteki001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A DEEP-DIVE INTO CRYPTOJACKING MALWARE: FROM AN EMPIRICAL ANALYSIS TO A DETECTION METHOD FOR COMPUTATIONALLY WEAK DEVICES

A dissertation submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Ege Tekiner

2021

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Ege Tekiner, and entitled A Deep-dive into Cryptojacking Malware: From an Empirical Analysis to a Detection Method for Computationally Weak Devices, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

[redacted]

Alexander Perez-Pons

[redacted]

Alex Afanasyev

[redacted]

Engin Kirda

[redacted]

A. Selcuk Uluagac, Major Professor

Date of Defense: November 3, 2021

The dissertation of Ege Tekiner is approved.

[redacted]

Dean John Volakis
College of Engineering and Computing

Andres G. Gil
Vice-President for Research and Economic Development
and Dean of University of Graduate School

Florida International University, 2021

DEDICATION

To my family, Serap Tekiner, Nejat Tekiner, Melis Tekiner

My Mentor, Engur Riza Pisirici

and my friends,

Dilara Hekimci-Adak, Hunter Adak, Akif Turker, Hakan Sekerci

ACKNOWLEDGMENTS

First of all, I would like to thank my major professor and advisor, Prof. Dr. A. Selcuk Uluagac, for his support, guidance and leadership during my research journey. His support and encouraging feedbacks lead me to higher success in my professional life.

I would also like to express my appreciation to the members of my dissertation committee, Dr. Alexander Perez-Pons, Dr. Alex Afanasyev, and Dr. Engin Kirda, for their valuable comments, guidance, and support.

Throughout my master's journey, I had a chance to work with several valuable researchers. I would like to thank Dr. Ali Aydin Selcuk, from TOBB University of Economics and Technology, Turkey and Dr. Engin Kirda from Northeastern University, USA for their great contribution, guidance and support on researches. I would like to thank my colleague and friend Dr. Abbas Acar for his support, and encouragement. I would also like to thank our post doctoral researcher Dr. Ahmet Aris, and all the members of Cyber-Physical Systems Security Lab (CSL) for everything.

Finally, I like to thank my parents and my sister who always support me, encourage me and never leave me alone even if they are in overseas during my master's. Finally, I would like to express my gratitude to Florida International University and U.S. National Science Foundation for financially supporting my research as well as the FIU community for providing a great environment for my research for this dissertation.

ABSTRACT OF THE DISSERTATION

A DEEP-DIVE INTO CRYPTOJACKING MALWARE: FROM AN EMPIRICAL ANALYSIS TO A DETECTION METHOD FOR COMPUTATIONALLY WEAK DEVICES

by

Ege Tekiner

Florida International University, 2021

Miami, Florida

Professor A. Selcuk Uluagac, Major Professor

Cryptojacking is an act of using a victim's computation power without his/her consent. Unauthorized mining costs extra electricity consumption and decreases the victim host's computational efficiency dramatically. In this thesis, we perform an extensive research on cryptojacking malware from every aspects. First, we present a systematic overview of cryptojacking malware based on the information obtained from the combination of academic research papers, two large cryptojacking datasets of samples, and numerous major attack instances. Second, we created a dataset of 6269 websites containing cryptomining scripts in their source codes to characterize the in-browser cryptomining ecosystem by differentiating permissioned and permissionless cryptomining samples. Third, we introduce an accurate and efficient IoT cryptojacking detection mechanism based on network traffic features that achieves an accuracy of 99%. Finally, we believe this thesis will greatly expand the scope of research and facilitate other novel solutions in the cryptojacking domain.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Since the day Bitcoin was released in 2009, blockchain-based cryptocurrencies have seen an increasing interest beyond specific communities such as banking and commercial entities. It has become so trivial and ubiquitous to conduct business with cryptocurrencies for any end-user as most financial institutions have already started to support them as a valid monetary system. Today, there are more than 2000 cryptocurrencies in existence. Especially in 2017, the interest for cryptocurrencies peaked with a total market value close to $1 trillion [Mar]. According to a recent Kaspersky report [Kasa], 19% of the world's population have bought some cryptocurrency before. However, buying cryptocurrency is not the only way of investing. Investors can also build mining pools to generate new coins to make a profit. Profitability in mining operations also attracted attackers to this swiftly-emerging ecosystem.

Cryptojacking is the act of using the victim's computational power without consent to mine cryptocurrency. This unauthorized mining operation costs extra electricity consumption and decreases the victim host's computational efficiency dramatically. As a result, the attacker transforms that unauthorized computational power into cryptocurrency. As of this writing, 32.3 million total cryptojacking attacks have been registered during the first half of 2020 [son20]. Such malicious cryptomining scripts were even found on some government websites around the world [UKG18]. Although in-browser cryptomining is instrumental for legitimate business purposes, the malicious or illegitimate usage is also gaining traction and is not unknown.

In-browser cryptomining allows websites to use their visitors' (i.e., clients') computational resources to mine cryptocurrency and to make revenue on behalf of the owner of a webpage. On the client side, in-browser mining is originally proposed as an alternative revenue mechanism to advertisements by the website owners, which in return, offer premium content or add-free surfing to its users. However, with the profitability of bitcoin and alternative cryptocurrencies, the attackers have started hijacking some popular websites [Min18] to embed cryptomining scripts (aka **cryptojacking**) and start mining without the knowledge and explicit consent of the users. Unfortunately, in the ecosystem of in-browser cryptomining, even though some website owners ask for explicit user consent before starting mining the clients' resources, none of the browser extensions, antivirus programs, and the detection studies in the literature differentiates the ones asking for explicit user consent (i.e., *permissioned*) from the ones starting mining without the knowledge and consent (i.e., *permissionless*) of the user. All in-browser cryptomining scripts are blacklisted and blocked by the prevention mechanisms. Google Chrome [New] and Opera [Dav] has recently announced that they would remove the cryptomining browser extensions from their web store and block the websites containing cryptomining scripts to protect their users as they are mostly being abused in practice.

Given the prevalent emerging nature of the cryptojacking malware, it is vital to detect and prevent unauthorized mining operations from abusing any computing platform's computational resources without the users' consent or permission. In this thesis we focus on three aspects of cryptojacking malware.

- Due to the seriousness of this emerging threat and the challenges presented above, many cryptojacking studies have been published before. However,these studies are either proposing a detection or preventio nmechanism against cryptojacking malware or analyzing the cryptojacking threat landscape. And, the literature lacks a systemic study covering both different cryptojacking malware types, techniques used by the cryptojacking malware, and a review of the cryptojacking studies in the literature.

- Unfortunately, in the ecosystem of in-browser cryptomining, even though some website owners ask for explicit user consent before starting mining the clients' resources, none of the browser extensions, antivirus programs, and the detection studies [AAUA21, ALB+19, CAA+19, ACA+17, NAB+21] in the literature differentiates the ones asking for explicit user consent (i.e., *permissioned*) from the ones starting mining without the knowledge and consent (i.e., *permissionless*) of the user. All in-browser cryptomining scripts are blacklisted and blocked by the prevention mechanisms. Google Chrome [New] and Opera [Dav] has recently announced that they would remove the cryptomining browser extensions from their web store and block the websites containing cryptomining scripts to protect their users as they are mostly being abused in practice.

  Indeed, the legitimate adoption of this emerging technology is instrumental for several reasons: First of all, today, a substantial portion of the revenue on the web is currently generated through online advertisements. However, the advertisement ecosystem is abused by the attackers, who redirect the users to malicious websites to spread the malware [ALUK19] (i.e., malvertising [SE11]) or ransomware [OALU21]. In this case, permissioned in-browser cryptomining would have been very beneficial by allowing the website owners to monetize their content by charging their users with their processing power instead of making without advertisements. This would reduce the risks posed by malicious advertisements. Second, permissioned in-browser cryptomining would have been a great mechanism to reach a large number of users and provide an easy payment method for nonprofit organizations and publishers.

  In this regard, there have already been a few attempts, such as the Hope project of UNICEF [UNIa] and the media outlet Salon [Sal19]. Third, in-browser cryptomining would offer convenience to end-users with its ad-free and customized content offered by the websites in exchange for uninterrupted use of the users' processing power.

Despite these potential benefits, the legitimate side of in-browser cryptomining has never been analyzed by the community due to its bad fame.

- Cryptojacking attackers initially targeted the personal computers by embedding the mining script inside the popular websites [TAU+21]. While the cryptojacking attacks gain more popularity and target bigger attack domains [IS, CS, Sea, CPR, Gre], IoT devices became the new favorite toy of attackers [IS, Cimc, Tre] because of their increasing usage in various different purposes such as healthcare, industry, and household, networks to provide different levels of connectivity to these environments. IoT devices are generally resource-constrained. Therefore, the attackers utilize techniques like botnet attacks to take control of them at scale and equip them to perform cryptocurrency mining on behalf of attacker.

  The famous IoT botnet malware, Mirai botnet [McM], performed massive Distributed Denial of Service(DDoS) attack in 2017, and some other [Dep, Tre, IoT] attacks followed the footsteps of the Mirai botnet. Moreover, the attacker(s) who performed Mirai botnet attack also used this network to mine bitcoin and turn the botnet network into a giant cryptojacking mining pool. Just recently, another Mirai-inspired botnet, LIQUOR IoT Botnet [IoT] started to mine Monero with its victims' IoT devices. While the IoT industry and capabilities of IoT devices continue broadening, it also gives attackers more space to widen their attacking surface.

1.1 Research Problems

This thesis has the following major research components and problems investigated:

1. A systematic overview of cryptojacking malware based on the information obtained from the combination of academic research papers, two large cryptojacking datasets of samples, and 45 major attack instances.

2. An empirical analysis with recent cryptomining samples focusing on the permissioned or authorized cryptomining.

3. A profit, usability, and user consent analysis on the existing cryptomining scripts provided by the service providers found in our dataset.

4. The introduction of a novel consent evaluation framework for the service providers and presented our benchmarking results for the 14 service providers we detected in the dataset.

5. The introduction of an accurate and efficient cryptojacking detection algorithm targeting IoT networks that is capable of detecting both in-browser and host-based cryptojacking malware

1.2 Contributions

The contributions of this thesis are as follows;

SoK: Cryptojacking Malware. In this study, to fill this gap in the literature, we present a systematic overview of cryptojacking malware based on the information obtained from the combination of 42 cryptojacking research papers, $\approx 26K$ cryptojacking samples with two unique datasets, and 45 major attacks instances. Given the widespread usage of cryptojacking, it is important to systematize the cryptojacking malware knowledge for the security community to accelerate further practical defense solutions against this ever-evolving threat.

In-Browser Cryptomining for Good: An Untold Story. In this contribution, we present a detailed empirical analysis of permissioned and permissionless in-browser cryptomining operations with real data collected from the web. Specifically, we created a dataset of **6269** unique websites containing cryptomining scripts in their source codes. Then, we performed a detailed cross-correlation analysis between the permissioned and permissionless cryptomining samples to reveal the differences (if any) between them and explore the characteristics of the permissioned cryptomining. In addition, we identified five different user consent methods used by the samples after a further analysis on the permissioned cryptomining samples.

Finally, based on the samples we analyzed, our findings, and service provider documentations, we revisit the permissioned cryptomining services with the following questions: 1) Can they be an alternative to advertisement? 2) Do they interrupt the users? 3) Do they satisfy consent requirements? We found that an affirmative answer is possible for each question if implemented properly by the service provider and the website owner, which led us to believe that the potential of permissioned in-browser cryptomining as a legitimate and viable monetization tool.

A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks. In this study, our focus is cryptojacking malware targeting IoT devices inside heterogeneous smart home networks. However, the detection of IoT cryptojacking is challenging because most of the IoT devices do not allow to be programmed to collect the hardware-level features while the network traffic-based features can be collected in a unified interface on the router, i.e., the devices do not have to be programmed or modified at all. Also, most of these devices are programmed in the cloud. Therefore, in this study, we used in-site network-based features to detect the IoT cryptojacking malware and propose an accurate, lightweight, and easy-to-implement cryptojacking detection system that can detect both kinds of cryptojacking attacks without interfering with the hardware or operating software of the devices inside the home network.

We performed extensive set of experiments to design and evaluate the best IoT cryptojacking detection mechanism. We first performed experiments to find the best-ranked features, the most accurate classifier, and the optimum training size. Then, we evaluated the effectiveness of our IoT cryptojacking detection mechanism with 12 novel experiments designed to assess various attacker behaviors and network settings. For this, we implemented the cryptojacking malware on IoT devices, a laptop, and a server in a safe setup.

1.3 Organization of Thesis

The rest of this thesis is organized as follows: In Chapter 2, we present pertinent background information that serves to support this thesis. In Chapter 3, the studies in the literature related to the work in this thesis are presented. Chapter 4 provides a systematic and detailed overview of cryptojacking malware. Then, in Chapter 5, we perform an empirical analysis of permissioned cryptojacking samples. In Chapter 6, we introduce a cryptojacking detection mechanism for computationally weak devics. Finally, we conclude the thesis and propose future research directions in Chapter 7.

CHAPTER 2

BACKGROUND

In this chapter, we provide background information pertinent to the context of the thesis. This includes a brief explanation of what the blockchain is and how it operates. Further, this section will provide necessary information regarding cryptomining and the differences between permissioned and permissionless cryptomining.

2.0.1 Blockchain

Blockchain is a distributed digital ledger technology storing the peer-to-peer (P2P) transactions conducted by the parties in the network in an immutable way. Blockchain structure consists of a chain of blocks. As an example, in Bitcoin [Nak08], each block has two parts: block header and transactions. A block header consists of the following information: 1) Hash of the previous block, 2) Version, 3) Timestamp, 4) Difficulty target, 5) Nonce, and 6) The root of a Merkle tree. By inclusion of the hash of the previous block, every block is mathematically bound to the previous one. This binding makes it impossible to change data from any block in the chain. On the other hand, the second part of each block includes a set of individually confirmed transactions.

2.0.2 Cryptocurrency Mining

The immutability of a blockchain is provided by a consensus mechanism, which is commonly realized by a ”Proof of Work” (PoW) protocol. The immutability of each block and the immutability of the entire blockchain are preserved thanks to the chain of block structure. In PoW, some nodes in the network solve a hash puzzle to find a unique hash value and broadcast it to all other nodes in the network. The first node broadcasting the valid hash value is rewarded with a block reward and collects transaction fees. A valid hash value is verified according to a difficulty target, i.e., if it satisfies the difficulty target, it is accepted by all other nodes, and the node that found the valid hash value is rewarded. Different PoW implementations usually have different methods for the difficulty target.

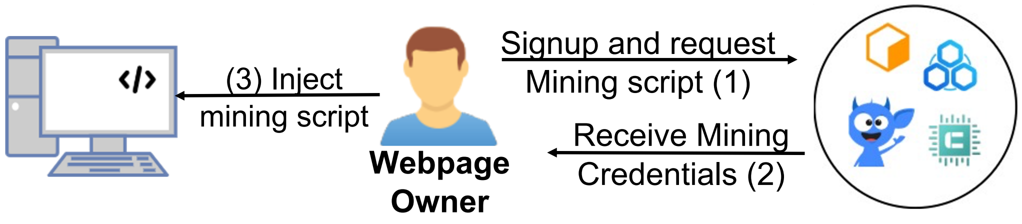
The miners try to find a valid hash value by trial-and-error by incrementing the nonce value for every trial. Once a valid hash value is found, the entire block is broadcast to the network, and the block is added to the end of the last block. This process is known as *cryptocurrency mining* (i.e., *cryptomining*), and it is the only way to create new cryptocurrencies. The chance of finding of valid hash value by a miner is directly proportional to the miner's hash power, which is related to the computational power of the underlying hardware. However, more hardware also increases electricity consumption. Therefore, attackers have an incentive to find new ways of increasing computational power without increasing their own electricity consumption.

Following the invention of Bitcoin, many other alternative cryptocurrencies (i.e., altcoins) have emerged and are still emerging. These new cryptocurrencies either claim to address some issues in Bitcoin (i.e., scalability [Kea20], privacy) or offer new applications (i.e., smart contracts [BDLF$^+$16]).

In the early days of Bitcoin, the mining was performed with the ordinary Central Processing Unit (CPU), and the users could easily utilize their regular CPUs for Bitcoin mining. Over time, Graphical Processing Unit (GPU)-based miners gained significant advantages over CPU miners as GPUs were specifically designed for high computational performance for heavy applications. Later, Field Programmable Gate Array (FPGA) have changed the cryptocurrency mining landscape as they were customizable hardware and provided significantly more profit than the CPU or GPU-based mining. Finally, the use of the Application-Specific Integrated Circuit (ASIC) based mining has recently dominated the mining industry as they are specially manufactured and configured for cryptocurrency mining.

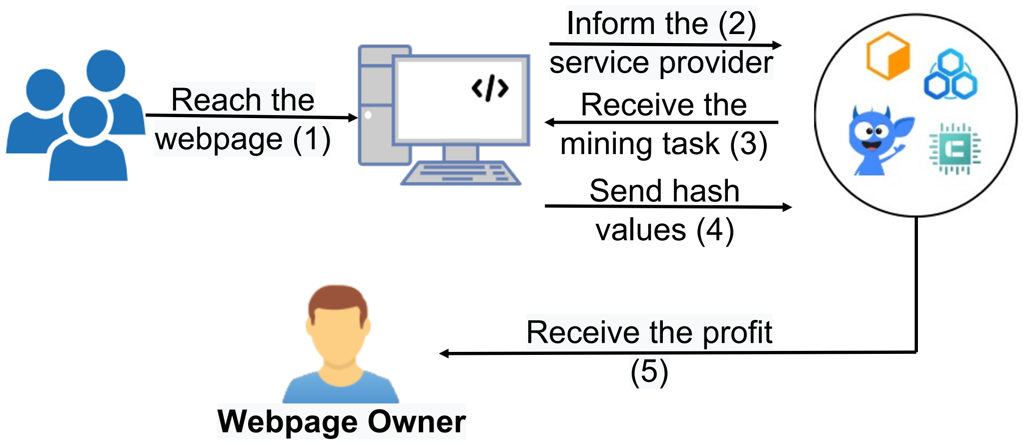
The alternative cryptocurrencies also used different hash functions in their blockchain structure, which led to variances in the mining process. For example, Monero [VS13] uses the CryptoNight algorithm as the hash function. CryptoNight is specifically designed for CPU and GPU mining. It uses L3 caches to prevent ASIC miners. With the use of RandomX [VS13] algorithm, Monero blockchain fully eliminated the ASIC miners and increased the advantage of the CPUs significantly. This feature makes Monero the only major cryptocurrency platform that was designed specifically to favor CPU mining to increase its spread. Moreover, Monero is also known as a private cryptocurrency, and it provides untraceability and unlinkability features through mixers and ring signatures. Monero's both ASIC-miner preventing characteristics and privacy features make it desirable for attackers.

Figure 2.1: Creation and injection of in-browser mining script



Service providers generally manage in-browser cryptomining source codes and operations, as shown in Figure 2.1. The webpage owner creates an account on the service provider's website and receives the needed script and credentials for the in-browser mining. The webpage owner embeds this code into the HTML source code or adds it as a plugin for some service providers. After this process, the in-browser mining operation starts as shown in Figure 2.2, and all the visitors become ad-hoc miners for these webpages and solve mining tasks for webpage owners. Mining tasks are assigned to the users by the service providers, or they may be directly coming from the mining

Figure 2.2: Lifecycle of in-browser mining



pool. At the end of the pre-defined period, the webpage owners receive the mining share after the service provider cuts the service commission. In this process, the users do not receive any profit.

2.0.3 Permissioned vs. Permissionless Cryptomining

We categorize the cryptomining scripts into two categories: 1) *permissioned* 2) *permissionless*. Permissioned cryptomining samples contain a code snippet for explicit user consent. In contrast, permissionless cryptomining samples do not ask for user consent, i.e., automatically starts mining without the visitor's knowledge or consent. However, while some service providers have methods to implement these options in their script, some of them have different ways of implementing such a user interface.

Listing 1 shows a sample in-browser cryptominer script provided by the Coinimp service provider [Coif], which is a currently active service provider. Line 5 in the code snippet contains a method for the user notification. The method can be used by adjusting the parameters by the website owners. The decision to include the notification or not is in the control of the website owner. If line 5 is not included, the miner will start automatically in the background without notifying the user before-

Listing 1: A sample permissioned in-browser cryptominer script.

```
<script src="<path-to-script>"></script>
<script>
    var _client = new Client.Anonymous("<site-key>", {throttle: 0, c:
        'w' });
    _client.start();
    _client.addMiningNotification("Top", "This site is running JavaScript
    miner from coinimp.com","#cccccc", 40, "#3d3d3d");
</script>
```

hand. Moreover, this method does not give the user the option to opt-out easily. We present the notification method provided by the Coinimp service provider here as a representative example, but there are other user consent methods provided by other service providers as well. We will analyze the service providers in Section 5.3.1 and different user consent types in Section 5.3.3 in further detail.

CHAPTER 3

LITERATURE REVIEW

The surge of cryptojacking malware, especially after 2017, also drew the attention of academia and resulted in many publications. We found these studies focus on three topics: 1) Cryptojacking detection studies, 2) Cryptojacking prevention studies, and 3) Cryptojacking analysis studies. Among 42 academic research papers, we found that 15 of them focus on the experimental analysis of the cryptojacking dataset. At the same time, 3 of them proposes a method for the detection and prevention of cryptojacking malware together, and 24 of them proposes only a method for the detection of the cryptojacking malware. In the next sub-sections, we give a review of these studies.

3.1 Cryptojacking Detection Studies

In this section, we survey the cryptojacking malware detection studies. Table 3.1 shows the list of the proposed cryptojacking detection mechanisms in the literature. The following sub-section gives a detailed overview of the dataset, platform, analysis method, features, and classifiers used in these detection mechanisms.

3.1.1 Dataset

A dataset is generally used to evaluate the effectiveness of the proposed detection method. Several datasets are commonly used in the cryptojacking malware detection literature. The most common one is Alexa top webpages [RP18, KMM+19, KBRS20, HYY+18, MWJR19, WFX+18]. Alexa sorts the most visited websites on the Internet; however, it does not provide the source code for these websites. Therefore, these studies also used Chrome Debugging Protocol to instrument the browser and collect the necessary information from the websites, except the study [WFX+18], which

Table 3.1: Cryptojacking malware detection mechanisms in the literature.

Ref	Dataset	Type	Method	Features	Classifier	Performance
Rüth et al. RZWH18	Three largest TLDs Alexa: 1M	In-browser	Static	Wasm signatures	SRSE	N/A
Minesweeper KVM+18	Alexa 1 Million	In-browser	Static	Wasm code CPU cache events	Matching	N/A
RAPID RP18	Alexa: 330.500	In-browser	Dynamic	Resource consumption (memory, network, and processor) and JavaScript API Events	SVM	Benign (best): Precision: 99.99% Recall: 99.99% F1: up to 99.99% Mining (best): Precision: 96.54% Recall: 95.48% F1: up to 96.0%
Muñoz et al. iMSVBR19	Network traffic of six cryptocurrencies using Stratum protocol	In-browser	Dynamic	Metadata of inbound and outbound network traffic	DT	Best: Accuracy: 99.9% Precision: 98.2% Recall: 90.7%
CapJack NWX+19	Five user applications and a Coinhive miner	In-browser	Dynamic	CPU utilization, Memmory, Disk read/write rate, Network interface)	CNN	87% Instant 99% After 11 seconds 98% Mobile Single 86% Mobile Cross 97% AWS single 89% AWS Cross
OUTGUARD KMM+19	Alexa 1M and 600K	In-browser	Dynamic	Js Execution Time, JS compilation Time, Garbage Collector, Iframe resource loads, CPU usage	SVM, RF	SVM (best): TPR: 97.9% FPR: 1.1%
CoinSpy KBRS20	100k websites from Alexa 1M and 50 manipulated cryptojacking websites	In-browser	Dynamic	CPU, Memory, Network behaviors	CNN	Accuracy:97%
MineCap NLFM20	The network traffic captured from two mining and streaming applications	In-browser	Dynamic	Network packages	IL	Accuracy: 98%, Precision: 99%, Recall: 97% Specifity: 99.9%
CMTracker HYY+18	Alexa 100k	In-browser	Dynamic	Hash and Stuck based profilers	Thr-based	100% TPR
Musch et al. MWJR19	Alexa 1M	In-browser	Dynamic	CPU usage	MA	N/A
Tahir et al. TDA+19	Manually created 320 non-mining and 100 mining websites	In-browser	Dynamic	HPC values	RF	Accuracy: 99.35%, Precision: 100%, Recall: 98%, AUC: 99%
SEISMIC WFX+18	500 webpages randomly selected from Alexa top 50K	In-browser	Dynamic	Wasm instructions	Matching	F1: 98%
MineThrottle BMZ20	Alexa 1M	In-browser	Dynamic	Block-level features CPU usage	Matching	FNR: 1.83%
Coinpolice PIB20	47k samples	In-browser	Dynamic	CPU usage, HPC, JS/WASM execution time and features, Throttling-independent timeseries	CNN	TPR:97.8 % FPR: 0.74%
Carlin et al. COSB18	Captured Opcode trace packets Virusshare (296 Samples)	In-browser	Dynamic	Opcodes	RF	TPR: 99.2 % FPR: 0.9, Precision: 99.2 Recall: 99.2
Liu et al. LZC+18	1159 samples collected from browsers' memory snapshot	In-browser	Dynamic	Heap snapshots Stack Features	RNN	Precision: 95, Recall: 93
Rauchberger et al. RSD+18	Alexa: 1M	In-browser	Dynamic	Web socket traffic	Matching	N/A
Caprolu et al. CRODP19	N/A	In-browser	Dynamic	Network traffic	RF,KFCV	TPR=92% ,FPR=0.8%
MINOS NAB+21	WASM Samples collected via PublicWWW	In-browser	Dynamic	Image frames of malicious samples	CNN	Accuracy: 98.97%
Yulianto et al. YSWAM19	PublicWWW and Blacklists	In-browser	Static and Dynamic	CPU usage	Matching	TPR:100%
CMBlock RS19	In-browser cryptojacking samples	In-browser	Static and Dynamic	Blacklists Behaviour	N/A	N/A
Gangwal et al. CGLP19	Combination daily user tasks and miners[1]	Host-based and In-browser	Dynamic	hardware events (e.g., branch-misses), software events (e.g., page-faults) hardware cache events (e.g., cache-misses)	RF, SVM	Recall: 97.84% Precision: 99.7% Accuracy:98.7%
Lachtar et al. LEBM20	N/A	Host-based and In-browser	Dynamic	CPU instructions	Matching	TPR:100 % FPR: ¡ 2%
Tanana et al. Tan20	40 In-browser and 10 executable-type cryptojacking	Host-based and In-browser	Dynamic	CPU utilization share RAM usage	N/A	TPR: 81%
Ahmad et al. ASKS19	Mixture of Benign and Malicious Network Packages	Host-based and In-browser	Dynamic	Network traffic	DCA	N/A
DeCrypto Pro MPB+20	1̃200 samples	Host-based and In-browser	Dynamic	HPC, CPU usage	k-NN, RF, LSTM	FPR: 2.5, Precision: 96, Recall: 97
Darabian et al. DHD+20	1500 active cryptomining collected from Virustotal in 2018	Host-based	Static and Dynamic	System calls, opcode sequences	RNN, CNN	System calls (best): LSTM: Accuracy:99% F1: 98% MCC: 98% FPR:0.6%
Crypto-Aegis CRODP19	Network traffic of 3 legitimate mining scripts and 3 daily user applications	Host-based	Dynamic	Packet sizes Interarrival times	RF	TPR:80-84% FPR: 0.9 - 1.2%

[1] The dataset was not available as of writing this paper (November 1, 2020). [2] Support Vector Machine: SVM, Random Forest: RF, Decision Tree: DT, Convolutional Neural Network: CNN, Recurrent Neural Network: RNN, Incremental Learning: IL, Threshold-based: Thr-based, Manual Analysis: MA, Dendritic Cell Algorithm: DCA, k-Nearest Neighbors: k-NN, Light-weight machine learning models: LSTM, Symantec RuleSpace Engine:SRSE, k-Fold Cross Validation:KFCV

works with a limited number (500) of websites. Moreover, the study in [KMM$^+$19] also used known and frequently updated blacklists [NoC, Mina, Coia] to build a ground truth for their training dataset, and then they performed an analysis using Alexa top 1M websites. In addition to the Alexa top websites, the study in [DHD$^+$20] used a cryptojacking dataset obtained from VirusTotal. They collected 1500 active Windows Portable Executable (PE32) cryptocurrency mining malware samples registered in 2018 and used the Cuckoo Sandbox [GTBS12] to obtain detailed behavioral reports on those samples. Furthermore, the studies in [CGLP19, iMSVBR19, NWX$^+$19, NLFM20] performed their analysis by installing the legitimate mining scripts, and the studies in [KBRS20, TDA$^+$19] manually injected miners to the websites to test their detection mechanisms.

3.1.2 Platform

Most of the cryptojacking detection mechanisms in the literature [RP18, CGLP19, NWX$^+$19, KMM$^+$19, KBRS20, NLFM20, HYY$^+$18, MWJR18, TDA$^+$19, WFX$^+$18, KVM$^+$18] are proposed for the detection of in-browser cryptojacking malware. There are only a few studies [DHD$^+$20, CRODP19] proposed for host-based cryptojacking malware. In addition, Conti et al. [CGLP19], propose a hardware-level detection mechanism, which can be used to detect both host-based and in-browser cryptojacking malware.

3.1.3 Analysis Features

As can be seen from Table 3.1, in the cryptojacking domain, the majority of the proposed detection methods are using dynamic analysis. The main reason for this is that mining scripts use a set of known instructions, and they follow and repeat predefined mining steps. For example, miners use cryptographic hash libraries and increment the value of a static variable (i.e., nonce) repeatedly or connect to some known service providers to continue to upload some output results and receive new tasks. These typical behaviors of the cryptojacking malware create a pattern and make them detectable by dynamic analysis. In the literature, only a few studies use static features such as opcodes [DHD+20] and WebAssembly (Wasm) instructions [KVM+18]. WebAssembly [HRS+17] is a low-level instruction format that allows programs to run closer to the machine-level language and provide higher performance via stack-based virtual machines [was]. This low-level instruction model lets the WebAssembly run the codes more efficiently, and this feature provides more profit because the cryptojacking script eliminates most of the delay caused by the code execution process. All major browsers in the market currently support WebAssembly.

Opcodes are machine language instructions that specify the operations to be performed and are used by system calls. The proposed detection system in [DHD+20] uses opcodes for static analysis, where opcodes are extracted using IDA Pro. In the cryptojacking example, opcodes focus on requests between mining scripts and the operating system's kernel. With this method, they achieve 95% accuracy with the Random Forest classifier.

On the other hand, many detection mechanisms have been proposed [RP18, NWX+19,CGLP19,KMM+19,KBRS20,NLFM20,HYY+18,MWJR18,TDA+19] using dynamics features. The most commonly used dynamic features in these studies are as follows:

- *CPU Events [RP18, NWX+19, KMM+19, KBRS20, MWJR18, YSWAM19, BMZ20, PIB20, LEBM20, Tan20, MPB+20]:* CPU events are the most commonly used features among the dynamic analysis-based detection mechanisms because in-browser cryptojacking scripts have to fetch the CPU instructions to perform the mining, independent of the used hardware. If an in-browser operation uses cryptographic libraries too frequently, which is abnormal for regular websites, it can be directly detected by CPU instructions. Even though CPU is the most crucial feature of cryptocurrency mining, using only CPU events as features may cause high false-positive rates (FPR) because flash gaming or online rendering websites also use the CPU of the system heavily for their operations. To keep FPR as low as possible, most detection methods use more than one features simultaneously [RP18, NWX+19, KBRS20, KVM+18, BMZ20, PIB20, Tan20, MPB+20].

- *Memory activities [RP18, CGLP19, NWX+19, KBRS20]:* Memory activity is another commonly used feature among the dynamic detection methods listed in Table 3.1.

Table 3.2: The list of publicly available blacklists.

Ref	Link
Nocoin [NoC]	`https://github.com/keraf/NoCoin`
CoinBlocker [Coia]	`https://zerodot1.gitlab.io/CoinBlockerListsWeb/index.html`
Minerblock [Mina]	`https://github.com/xd4rker/MinerBlock/blob/master/assets/filters.txt`
Coinhive Blocker [Coib]	`https://raw.githubusercontent.com/Marfjeh/coinhive-block/master/domains`
Andreas CH Blocker [and]	`https://raw.githubusercontent.com/andreas0607/CoinHive-blocker/master/blacklist.json`

- *Network package [RP18, iMSVBR19, NWX+19, KBRS20, NLFM20, CRODP19]:* Network packages are also a handy and useful method to detect cryptojacking activity because of the massive network traffic generated while uploading the calculated hash values to the service provider. The studies [RP18, iMSVBR19, NWX+19, KBRS20] utilized network traffic rate as an additional feature along with other features such as memory and CPU-related features. On the other hand, the studies in [NLFM20, CRODP19] used only network packages for cryptojacking malware

Table 3.3: The list of open-source cryptojacking malware detection implementations.

Ref	Implementation Link	Description	Last Update
CMTracker [HYY+18]	`https://github.com/deluser8/cmtracker`	code	Sep 21, 2018
Minesweeper [KVM+18]	`https://github.com/vusec/minesweeper`	data and code	Mar 17, 2020
OUTGUARD [KMM+19]	`https://github.com/teamnsrg/outguard`	data and code	Sep 6, 2019
SEISMIC [WFX+18]	`https://github.com/wenhao1006/SEISMIC`	code	Sep 10, 2019
Retro Blacklist [HPV+20]	`https://github.com/retrocryptomining/`	data and code	Jul 16, 2020

detection. Particularly, Neto et al. [NLFM20] use the network flow as a feature, while Caprolu et al. [CRODP19] use interarrival times and packet sizes as features in their detection algorithm.

- *JavaScript (JS) compilation and execution time [KMM+19, PIB20]:* In [KMM+19, PIB20], it has been shown that JS engine execution time and JS compilation time is significantly affected by cryptojacking malware. However, online games and other online rendering platforms can also cause the same behavior causing false positives in the detection mechanism. Therefore, the study in [KMM+19] also uses CPU usage, garbage collector, and iframe resource loads as secondary features to obtain more accurate results and decrease false positives. The garbage collector is a feature of the JS programming language to optimize memory usage, and it deletes unnecessary data from memory and prevents memory overloading. The memory and CPU continuously interact with each other during the mining operation, and the CPU sends calculated data to the memory. The garbage collector deletes all calculated hash values one by one after being sent to the service provider; therefore, the mining process causes irregular usage of the garbage collector. Due to this behavior, the garbage collector can be used as a feature for the dynamic detection mechanism. Iframes are the HTML tags used for embedding another program/-function to an HTML source code. Mining scripts are inserted into those tags and work under HTML codes. Similar to previous features, cryptojacking scripts cause irregular usage in iframe resource loads. This feature cannot be used as a primary feature because too many modern web applications use iframe resources irregularly, and it may cause a high false-positive rate.

- *Hardware Performance Counter (HPC) [TDA+19, PIB20, MPB+20]:* HPC values [DWA+19] are used on modern computers' CPUs and keepthe record of internal CPU events (e.g., Cycles, Cache misses). The values of the registers with

CPU clock cycles and executed instructions provide unique information about the behaviors of a running application. Several studies check the hardware activities and the related applications with HPC values to detect the cryptocurrency mining operations on the system.

- *System calls [DHD+20]:* System calls are the API structures that enable the connection between applications and the running system's kernel. System calls run with level 0 privileges to invoke calls and request services from the OS's kernel. The proposed detection system in [DHD+20] uses the system calls for dynamic analysis, and system calls are recorded using the Cuckoo Sandbox. Then, the system calls are used to train deep learning models, and they achieve 99% accuracy.

3.1.4 Classifier and Performance

The collected features are mostly used to train different machine learning classifiers such as Support Vector Machine (SVM) [RP18, CGLP19, KMM+19], Random Forest [CGLP19, TDA+19], Neural Network [KBRS20, DHD+20], Decision Tree [iMSVBR19]. Moreover, Neto et al. [NLFM20] proposed the use of incremental learning, which takes the classification probabilities of an ensemble of classifiers as a feature for an incremental learning process. Moreover, Hong et al. [HYY+18], proposed a threshold-based detection, and the studies in [WFX+18, KVM+18] used a static matching method to detect certain functions in the script. Musch et al. [MWJR19] only report the number of detected websites in the Top 1M Alexa websites. As can be seen from Table 3.1, all classifiers achieve a near-perfect ($\sim$100%) detection results.

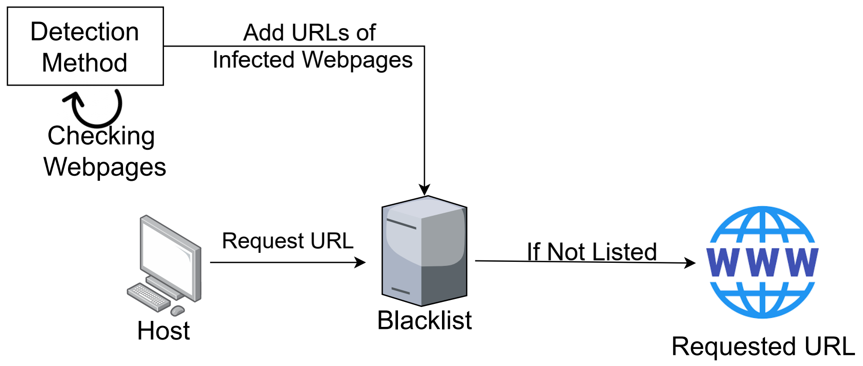
3.1.5 Open Source Implementations

Finally, some of the studies [HYY+18, KVM+18, KMM+19, WFX+18, Ret] published their code to help the research community.Table 3 presents the list of open-source cryptojacking malware implementations.

3.2 Cryptojacking Prevention Studies

A majority of the detection mechanisms do not focus on preventing or interrupting of cryptojacking malware; however, there are still several studies [YSWAM19, BMZ20, RS19] focusing on both the detection and prevention of cryptojacking malware. Using dynamic features to detect ongoing cryptojacking is like other dynamic analysis studies, but their prevention methods vary. While Yulianto et al. [YSWAM19] only raises a notification, Bian et al. [BMZ20] sleep the mining process, and Razali et al. [RS19] directly kill the related process.

Figure 3.1: Blacklisting method.



For cryptojacking prevention, there are also several tools in the market. Against host-based cryptojacking malware, proprietary antivirus programs [Ava, Nor][1] are commonly preferred. Against in-browser cryptojacking malware, open-source browser extensions such as NoCoin [NoC] and MinerBlock [Mina] are widely used. These open-source browser extensions are based on blacklisting, where the lists are updated as

[1]As these programs are closed-source, their methods are not publicly available.

Table 3.4: Cryptojacking malware analysis studies in the literature.

Ref	Cryptojacking Dataset	Sample Type	Focus of the Study
[HDM+14]	2000 executable	binary	the practice of using compromised PCs to mine Bitcoin
[ELMC18]	33282 websites	script	prevalence analysis
[Sig18]	-	-	how cybercriminals are exploiting cryptomining
[BBD19a]	5190 websites	script	campaign and domain analysis
[MJS19]	XMR-stak, cpuminer-multi	binary	attack impact on consumer devices and user annoyance
[SKM19]	5700 websites	script	static, dynamics and economic analysis
[ZWM19]	CoinHive cryptominer	script	sample characteristics and network traffic analysis
[PST19]	1.2 million miners	binary	currencies, actors , campaign and earning analysis, underground markets
[PIM19]	107511 websites	script	profitability and the imposed overheads
[BBD19b]	3.2 TB historical scan results	script	investigation of a new type of attack that exploits Internet infrastructure for cryptomining
[CBOS19]	-	-	business model, threat sources, implications, mitigations, legality and ethics
[ANDB20]	53 websites	script	sample characteristics
[VGOB20]	2770 websites	script	activeness analysis
[ZWMO20]	XMRig miner	binary	sample characteristics
[HPV+20]	156 domains, 25892 proxies	script	impact on the web users

new malicious domains are discovered. Table 3.2 shows the list of publicly available blacklists that we identified during our research. Browser extensions warn the user when the user wants to access a website on the blacklist. Figure 3.1 shows the blacklisting process, which is repeated as a continuous loop.

Pure blacklisting-based prevention is not an efficient way for stopping cryptojacking malware because attackers can easily change their domain by domain fluxing or other methods to downshift the effects of blacklists. There are also some new methods [RMY20] proposed by researchers for better and more optimized blacklisting, but even dynamic blacklisting methods are not fully effective nor protective [YRRR12] against domain fluxing methods.

3.3 Cryptojacking Analysis Studies

In addition to the cryptojacking malware detection and prevention studies, some researchers also performed empirical measurement studies to understand the cryptojacking threat landscape better. Table 3.4 shows cryptojacking malware analysis studies in the literature. In these studies, cryptominers are either in the format of binary [MJS19, ZWMO20, PST19, HDM+14] or script [BBD19a, HPV+20, SKM19, VGOB20, ELMC18, ZWM19, ANDB20, PIM19] except [CBOS19, Sig18] where the findings in these studies are based on the other studies and publicly available documents.

Researchers analyzed several different perspectives of cryptojacking. In the first study [HDM$^+$14], the authors analyze the binary samples identified as engaged in mining operations to characterize their scope, operations, and revenue. This is the first and only study analyzing Bitcoin miners, where the samples used in other studies are mining Monero. The increase in the cryptojacking malware attack instances in 2017 also drew researchers' attention. [ELMC18] is the first study analyzing the Monero cryptojacking samples, where the authors used over 30000 websites utilizing *coinhive.min.js* library for the prevalence analysis of cryptojacking samples. Many follow-up studies are published. For example, the studies [ZWM19, ANDB20, ZWMO20] also performed an analysis of the cryptojacking samples to identify characteristics of the samples. In addition, the studies in [MJS19, HPV$^+$20] performed the impact analysis. Particularly, [MJS19] analyzed the attack impact on consumer devices and user annoyance, and [HPV$^+$20] analyzed the impact of cryptojacking malware on web users, while [PIM19] analyzed the overhead of cryptojacking samples. In an interesting study, the authors in [BBD19b] investigated a new type of attack exploiting the Internet infrastructure for cryptomining, which indeed has an impact on 1.4M infected routers.

Moreover, there are also studies performing the economic analysis of cryptojacking samples such as [SKM19, PST19, PIM19]. Other than that, the authors in [BBD19a, PST19] performed a campaign analysis of the cryptojacking samples and [VGOB20] analyzed the activeness of cryptojacking threat after the discontinuation of Coinhive. Finally, while [Sig18] gives an overview of how cybercriminals are exploiting cryptomining, [CBOS19] presents a review of the business model, threat sources, implications, mitigations, legality, and ethics of cryptojacking malware.

CHAPTER 4

A SYSTEMATIC OVERVIEW OF CRYPTOJACKING MALWARE

4.1 Introduction

Emerging blockchain and cryptocurrency-based technologies are redefining the way we conduct business in cyberspace. Today, a myriad of blockchain and cryptocurrency systems, applications, and technologies are widely available to companies, end-users, and even malicious actors who want to exploit the computational resources of regular users through *cryptojacking* malware. Especially with ready-to-use mining scripts easily provided by service providers (e.g., Coinhive) and untraceable cryptocurrencies (e.g., Monero), cryptojacking malware has become an indispensable tool for attackers. Indeed, the banking industry, major commercial websites, government and military servers (e.g., US Dept. of Defense), online video sharing platforms (e.g., Youtube), gaming platforms (e.g., Nintendo), critical infrastructure resources (e.g., routers), and even recently widely popular remote video conferencing/meeting programs (e.g., Zoom during the Covid-19 pandemic) have all been the victims of powerful cryptojacking malware campaigns. Nonetheless, existing detection methods such as browser extensions that protect users with blacklist methods or antivirus programs with different analysis methods can only provide a partial panacea to this emerging cryptojacking issue as the attackers can easily bypass them by using obfuscation techniques or changing their domains or scripts frequently. Therefore, many studies in the literature proposed cryptojacking malware detection methods using various dynamic/behavioral features. However, the literature lacks a systemic study with a deep understanding of the emerging cryptojacking malware.

To fill this gap in the literature, in this chapter, we present a systematic overview of cryptojacking malware based on the information obtained from the combination of academic research papers, two large cryptojacking datasets of samples, and 45 major attack instances.

In-browser cryptojacking examples. In a major attack, cryptojacking malware was merged with Google's advertisement packages on Youtube [Min18]. The infected ads package compiled by victims' host performed unauthorized mining as long as victims stayed at the related page. Youtube and similar media content providers are ideal for the attackers because of their relative trustworthiness, popularity, and average time spent on those webpages by the users. In another incident, cryptojacking malware was found in a plugin provided by the UK government [UKG18]. At the time, this plugin was in use by several thousands of governmental and non-governmental webpages.

Cryptojacking examples found on critical servers. In addition to cryptojacking malware embedded into webpages, cryptojacking malware has also been found in *well-protected governmental and military servers.* The USA Department of Defense discovered cryptojacking malware in their servers during a bug-bounty challenge [DOD]. The cryptojacking malware found in the DOD servers was created by the famous service provider Coinhive [Coic] and mined 35.4 Monero coin during its existence. Similarly, another governmental case came up from the Russian Nuclear Weapon Research Center [Mil]. Several scientists working at this institution were arrested for uploading cryptocurrency miners into the facility servers. Moreover, attackers do not only use the scripts provided by the service providers but also modified the non-malicious, legitimate, open-source cryptominers. For example, a cybersecurity company detected an irregular data transmission to a well-known European-based botnet from the corporate network of an Italian bank [Won]. Further investigation identified that this malware was, in fact, a Bitcoin miner.

Cryptojacking examples utilizing advanced techniques. There have also been many incidents where the attackers used *advanced techniques* to spread cryptojacking malware. For example, in an incident, a known botnet, Vollgar, attacked all MySQL servers in the world [Cimb] to take over the admin accounts and inject cryptocurrency miners into those servers. Another recent incident was reported for the Zoom video conferencing program [Ole] during the peak of the Covid-19 pandemic, in which the attacker(s) merged the main Zoom application and cryptojacking malware and published it via different file-sharing platforms. In other similar incidents, attackers used gaming platforms such as Steam [Ken] and game consoles such as Nintendo Switch [Smi] to embed and distribute cryptojacking malware. Last but not least, in a recent study [BBD19b], researchers discovered a firmware exploit in Mikrotik routers that were used to embed cryptomining code into every outgoing web connection, where 1.4 million MikroTik routers were exploited.

Challenges of cryptojacking detection. Given the prevalent emerging nature of the cryptojacking malware, it is vital to detect and prevent unauthorized mining operations from abusing any computing platform's computational resources without the users' consent or permission. However, though it is critical, detecting cryptojacking is challenging because it is different from traditional malware in several ways. First, they abuse their victims' computational power instead of harming or controlling them as in the case of traditional malware. Traditional malware detection and prevention systems are optimized for detecting the harmful behaviors of the malware, but cryptojacking malware only uses computing resources and sends back the calculated hash values to the attacker; so the malware detection systems commonly consider cryptojacking malware as a heavy application that needs high-performance usage. Second, they can also be used or embedded in legitimate websites, which makes them harder to notice because those websites are often trustworthy, and users do not expect any nonconsensual mining on their computers. Third, while in traditional malware attacks, the attacker may ultimately target to exfiltrate sensitive information (i.e., Advanced Persistent Threat (APT)), to make the machine unavailable (i.e., Distributed Denial of Service (DDoS)) or to take control of the victim's machine (i.e., Botnet), in cryptojacking malware attacks, the attacker's goal is to stay stealthy on the system as long as possible since the attack's revenue is directly proportional to the time a cryptojacking malware goes undetected. Therefore, attackers use filtering and obfuscation techniques that make their malware harder for detection systems and harder to be noticed by the users.

4.2 Cryptojacking Malware Types

Cryptojacking malware, also known as cryptocurrency mining malware, compromises the computational resources of the victim's device (i.e., computers, mobile devices) without the authorization of its user to mine cryptocurrencies and receive rewards. A cryptojacking malware's lifecycle consists of three main phases: *1) script preparation, 2) script injection,* and *3) the attack.* The script preparation and attack phases are the same for all cryptojacking malware types. In contrast, the script injection phase is conducted either by injecting the malicious script into the websites or locally embedding the malware into other applications. Based on this, we classify the cryptojacking malware into two categories: *1) In-browser cryptojacking* and *2) Host-based cryptojacking.* In the following sub-sections, we explain the lifecycle of both in-browser and host-based cryptojacking malware.

4.2.1 Type-I: In-browser Cryptojacking

The development of web technologies such as JavaScript (JS) and WebAssembly (Wasm) enabled interactive web content, which can access the several computational resources (e.g., CPU) of the victim's device (e.g., computer or mobile device). In-browser cryptojacking malware uses these web technologies to create unauthorized access to the victim's system for cryptocurrency mining via web page interactions on the victim's CPU.

Figure 4.1: Script preparation and injection phases of a in-browser cryptojacking malware.

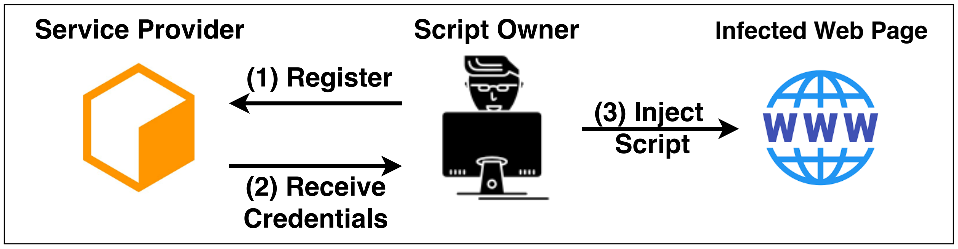
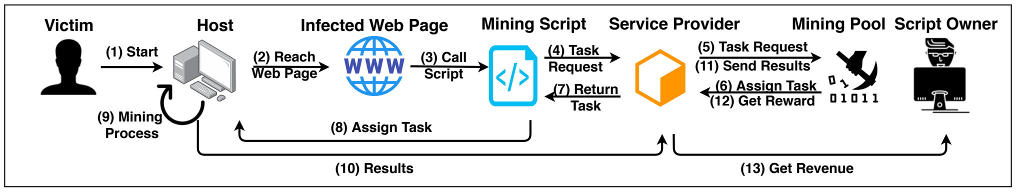


Figure 4.1 shows the script preparation and injection phases of in-browser cryptojacking malware. The script owner[1] first registers (Step 1) and receives its service credentials and ready-to-use mining scripts from the service provider (Step 2). The service provider separates the mining tasks among its users and collects all the revenue from the mining pool later to be shared among its users. After receiving the service credentials, the script owner injects the malicious cryptojacking script into the website's HTML source code (Step 3). We explain this and other cryptojacking infection methods in Section 4.3.2 in detail.

In the attack phase, as shown in Figure 4.2, victims first reach the website source code from their devices (Step 1,2). The web browser loads the website and automatically calls the cryptojacking mining script (Step 3). Once the script is executed,

[1]We call it script owner rather than an attacker because the script can also be used for legitimate purposes.

Figure 4.2: The lifecycle of a in-browser cryptojacking malware.



it requests a mining task from the service provider (Step 4). The service provider transfers the task request to the mining pool (Step 5). Then, the mining pool assigns the mining task (Step 6). The service provider returns the task to the mining script (Step 7). The mining script returns this new mining assignment to the victim's computer (Step 8), and the victim's device starts the mining process (Step 9). As long as the mining script and service provider remain online, the script continues the mining process on the victim's computer (Step 9) and then returns the mining results to the service provider (Step 10) directly. The service provider collects all the data from different sources and sends the results to the mining pool (Step 11). Finally, the mining pool sends the reward back to the service provider in the form of a mined currency (Step 12). The script owner receives its share from the service providers using its service credentials after the service provider cuts its service fee. In this ecosystem, the attackers use the CPU power of their victims, and the victims do not receive any payment nor benefit from any other entity.

4.2.2 Type-II: Host-based Cryptojacking

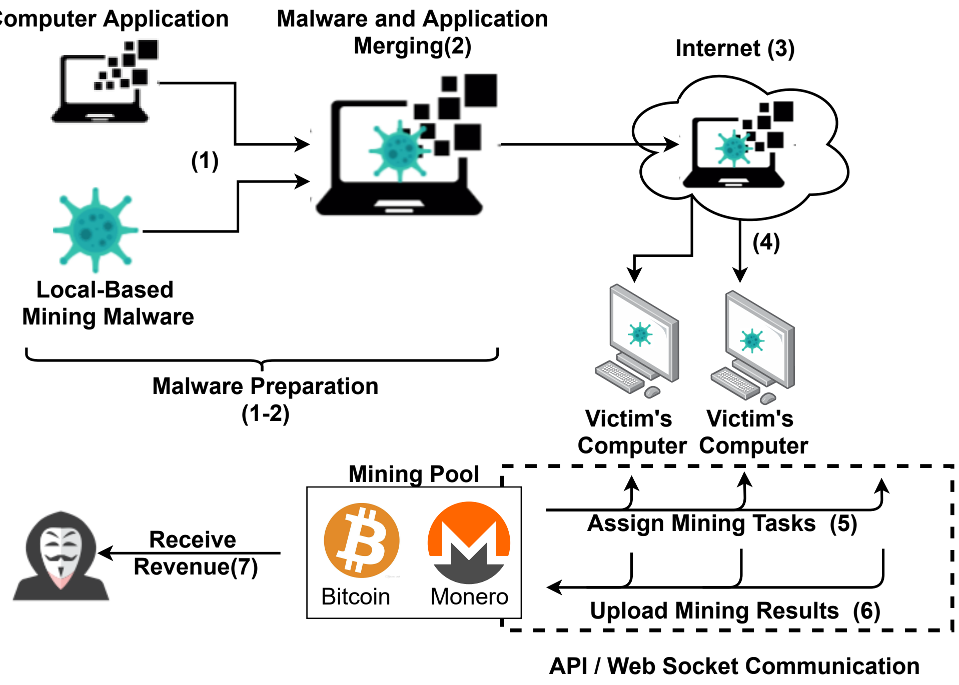
Host-based cryptojacking is a silent malware that attackers employ to access the victim host's resources and to make it a zombie computer for the malware owner. Compared to in-browser cryptojacking malware, host-based malware does not access the victim's computation power through a web script; instead, they need to be installed on the host system. Therefore, they are generally delivered to the host system through methods such as embedded into third-party applications [Ole, San], using vulnerabilities [MV], or social engineering techniques [McD], or as a payload in the drive-by-download technique [RS]. We explain these methods in more detail in Section 4.3.2.

Figure 4.3 shows the lifecycle of a host-based cryptojacking malware. The script preparation phase starts with the creation of unauthorized cryptocurrency mining malware (1). Then, the attacker merges this malware with a legitimate application to trick the victim (2). After the malware preparation, the malware injection process starts with uploading this malicious application to online data-sharing platforms (e.g., torrent, public clouds) (3). When the victim downloads any of the infected applications and installs them on their host machines (e.g., Personal Computer, IoT device, Server)(4), the malware injection phase of the lifecycle is completed.

In the attack phase, the host-based cryptojacking malware is connected to the mining pool via web socket or API and receives the hash puzzle tasks to calculate hash values (5). The calculated hash values are sent back to the mining pool (6). Finally, the attacker receives all of the revenue without any energy consumption (7) and not sharing anything with the victim.

After receiving all its revenue in the form of cryptocurrency from the service provider, the attacker has three options to use its revenue: 1) Converting to fiat currency via exchanges or p2p transactions, 2) Using it as a cryptocurrency for a service [LYK+19], or 3) Using cryptocurrency mixing services [GJPS18, BNM+14] to cover its traces. Further end-to-end analysis of the cryptojacking economy/payments is out of this study's scope, and similar studies can be found in the ransomware domain [HAL+18, PCHD19, CGR18, KRB+15].

Figure 4.3: The lifecycle of a host-based cryptojacking malware.


Computer Application
Malware and Application Merging(2)
Internet (3)
(1)
(4)
Local-Based Mining Malware
Malware Preparation (1-2)
Victim's Computer
Victim's Computer
Mining Pool
Receive Revenue(7)
Bitcoin
Monero
Assign Mining Tasks (5)
Upload Mining Results (6)
API / Web Socket Communication

4.3 Cryptojacking Malware Techniques

In this section, we explain the techniques used by cryptojacking malware. Particularly, we articulate on the following:

- Source of cryptojacking malware
- Infection methods
- Victim platform types
- Target cryptocurrencies
- Evasion and obfuscation techniques

4.3.1 Source of Cryptojacking Malware

This sub-section explains whom the scripts are created by and how they are distributed to attackers.

Service Providers

The service providers are the leading creators and distributors of cryptojacking scripts. The service providers give every user a unique ID to distinguish them in terms of the hash power. The service provider generates the script for the user regardless of the user is malicious or not. All the user needs to do is copy and paste the script to create a malicious sample for the attack.

Coinhive [Coic] was the first service provider to offer a ready-to-use in-browser mining script in 2017 to create an alternative income for web site and content owners. Even though the initial idea of Coinhive was to provide an alternative revenue to webpage owners, it rapidly became popular among attackers. During the operation of Coinhive, they were holding a significant share of the total hash rate of Monero. After the sharp decrease in Monero's price [Mona], Coinhive was shut down by their owners in March 2019 due to the business' being no longer profitable.

Some of the alternative service providers which had continued/continuing their operations are Authedmine [Coid], Browsermine [Bro], Coinhave [Coie], Coinimp [Coif], Coin nebula [Coig], Cryptoloot [crya], DeepMiner [Dee], JSECoin [JSE], Monerise [Monc], Nerohut [Ner], Webmine [webd], WebminerPool [Webg], and Webminepool [webe]. Some of these service providers also came up with several new functionalities, such as offering a user notification method or a GUI for the user to adjust the cryptomining parameters. Note that, we also verified these service providers using the samples in the PublicWWW dataset. In order to find the corresponding service providers of each sample, we performed a keyword search on the HTML source code of all samples. We found that 5328 samples use one of these 14 aforementioned service providers, while 941 samples with unknown service providers. Moreover, we also found out that 144 samples are using scripts from multiple service providers in their source codes. More details on the PublicWWW dataset can be found in Appendix.

Cryptominer Software

Blockchain networks rely on several network protocols and cryptographic authentication methods. Miners must be part of these protocols and follow the rules provided and developed by the communities. PoW-based cryptocurrencies also have specific rules for their blockchain networks. Due to blockchain technology's public and open nature, the source code of these miners are published by the communities via code sharing and communal development platforms. Attackers can easily obtain and modify these miners and adopt them to perform mining inside their victims' host machines. Moreover, there are also several plug-and-play style mining applications provided by several mining pools. Attackers are also modifying these applications for cryptojacking. For example, XMRig [XMR] is a legitimate high-performance Monero miner implementation, and it is open-source. Its signature is found in several highly impactful attacks affecting millions of end devices around the world [Gru, Kasb], which are also reported by Palo Alto Networks and IBM. Moreover, we also found 139 unique samples that are labeled with the signature of "xmrig" in our VT dataset.

4.3.2 Infection Methods

In this section, we explain the infection methods used by cryptojacking malware in detail.

Website Owners

Website owners, who have admin access to the website's servers, may employ in-browser mining scripts to gain extra revenue or provide in exchange of an alternative option to premium content they provide. Only with this method, webpage owners may receive the revenue of the script in their webpage. While some website owners inform their visitors about the cryptomining script they employ, some others do not inform their visitors, and this behavior can be considered as crime [Parb] in several countries.

Compromised Websites

Attackers may inject their cryptojacking malware into random web pages that have several vulnerabilities. Indeed the name cryptojacking itself is the combination of ”cryptomining” and ”hijacking.” Ruth et al. [RZWH18] state that ten different users created 85% of all Coinhive scripts they found. The owners of these webpages do not have any information about these scripts; additionally, they do not profit from them.

Several works claim that the attackers generally use the same ID for all the infected web pages, making them more traceable. For example, the authors in [BBD19a] reveals the cryptojacking campaigns through this method and discover that most of these campaigns utilize the vulnerabilities such as remote code execution vulnerabilities. When we investigated the common instances related to this domain, one security company found cryptojacking malware inside of the Indian government webpages [Chr], which affect all *ap.gov.in* domains and sub-domains.

Malicious Ads

Some attackers embed their cryptojacking malware into JavaScript-based ads and distribute them via mining scripts. With this method, the attackers can reach random users without any extra effort. To make this attack, they do not need to infect any webpage or application. YouTube [Min18] and Google ad [Cla] services were also infected and the users of these websites and their services became the victims of the cryptojacking attacks. The attackers successfully mined Monero with their visitors. The attackers successfully mined Monero with their visitors. The advantage of this method is that it allows attackers to reach a large number of visitors when it is embedded into popular websites without getting access to the website's servers.

Malicious Browser Extensions

Browser extensions can also reach the computer's CPU sources and act like cryptojacking malware located into a webpage. These extensions have a major distinctive difference; they can stay online and perform mining as long as the infected browser remains open independent from the websites accessed by the victim. However, major browser operators like Google announced that they would ban all the cryptomining extensions on their platform regardless of their intention as it is mostly being abused in practice [New].

Third-party Software

Merging malware with any market application and publishing it via several sharing platforms is a well-known method among the attackers to spread the malware. Attackers modify the cryptominer software to run cryptojacking in the background and merge it with legitimate applications. The attackers tend to use computation-intensive applications (e.g., animation applications, games with high hardware needs, engineering programs) because the use of those applications means that the victims' system has computationally powerful hardware and the application that host-based cryptojacking malware embedded, have access permission to the needed hardware components of the victim's host system.

Several major instances have already happened, such as, one attacker merged Zoom [Ole] video calling application with a regular bitcoin miner and distributed it via several sharing platforms. In another attack, the attackers used a popular video game Fortnite to spread the virus [Pea] to mine Bitcoin. Unlike the in-browser mining, which became popular in 2017, we found the attack instances using this method even in 2013, where the Bitcoin mining script found as part of the game's code itself [SC].

Exploited Vulnerabilities

In several cases, attackers exploit several zero-day vulnerabilities that they found in hardware and software. Attackers inject their mining malware into several devices and make them mine cryptocurrency. There are several important instances happened in the last several years. The most remarkable example directly affects 1.4 Million Mikrotik [BBD19b] routers globally, and a vulnerability in the hardware operating system causes this instance. The researchers claim that a major percentage of Remote Code Execution (RCE) attacks [Avi] aims to locate mining scripts inside the host systems.

Social Engineering Techniques

Social engineering is a commonly used technique among malware attackers to by-pass security practices. Similarly, attackers also use social engineering attacks to manipulate human psychology and navigate the victims' access or install malicious software on their computers. The researchers have observed that attackers are still using old techniques such as social engineering to install cryptojacking malware on their victims' computers [Sch].

Drive-by Download

A drive-by download is another technique used by malware attackers to deliver and install malicious files to victims' devices without their knowledge. Victims may face this attack while visiting a web page, opening a pop-up window, or checking an email attachment. In one case [RS], the attackers used this method to inject their cryptojacking malware into their victims' devices. They exploited shell execution vulnerability to download their cryptojacking malware to victims' computers directly.

4.3.3 Victim Platform Types

Browser

Browsers are the most commonly used victim platforms as the attackers do not need to deliver any malicious payload to the victim to use the computational resources of the victim. In other words, when the victim reaches the infected webpage, the malware automatically starts mining and do not leave any data behind. The second significant advantage of the browser environment is, thanks to service providers, ready-to-use mining scripts can be applied to any webpage very easily and quickly. The studies in the literature that we also present in Section 6 mostly focus on in-browser cryptojacking. However, the attackers can access only the CPUs of the victims through the browsers, which makes them infeasible for the currencies allowing ASIC miners such as Bitcoin. Therefore, cryptojacking malware samples utilizing browsers mostly mine Monero or other cryptocurrencies, which allow cryptomining by personal computers on non-ASIC CPU architectures.

Personal Computers

Personal computers are generally designed to allow end-users to perform their daily tasks. Personal computers are recently modified to overcome high-level computations to allow their users to use heavy-computation applications (e.g., video-games, video rendering applications). Attackers targeting personal computers aim to reach many victims because a limited number of victims would not be profitable. In-browser cryptojacking embedded into popular websites is ideal for this type of cryptojacking attack. In addition, they can also instantiate such an attack through large-scale campaigns. For example, in [Win], Cisco researchers document their findings of a two-year campaign delivering XMRig in their payload. They also observed that the malware "makes a minimal effort to hide their actions" and posting the malware "on online forms and social media" to increase the victim pool.

On-premise Server

On-premise (i.e., in-house) servers are the servers where the data is stored and protected on-site. It is preferred by highly critical organizations such as governmental organizations as it offers greater security and full control over the hardware and data.

However, on-premise servers are also another victim platform type attacked by the host-based cryptojacking malware samples. Compared to personal computers, on-premise servers are more computationally powerful and host numerous services accessed by many connections. This allows attackers to the broader attack surface. Still, the attackers have to find a way to deliver and install the cryptomining script on the on-premise server to access this platform. In several instances, the attackers used system vulnerabilities [Won], third party infected software [UKG18], and several social engineering methods [Sch] to install cryptojacking malware to the victims' on-premise server.

Cloud Server

Cryptojacking malware also exploits cloud resources to mine cryptocurrencies. Cloud-based cryptojacking attack is a fast-spreading problem in the last two years, where it became popular, especially after the shutdown of the Coinhive when the attackers were looking for new platforms to infect. Attackers target several vulnerabilities to hijack victims' cloud servers and locate cryptocurrency miners into their systems.

Clouds servers, especially Infrastructure-as-a-service platforms such as Amazon Web Services (AWS), are being targeted by the attackers because of their:

- Virtually infinite resources,
- Large attack surface due to server structure,
- Malware spreading capabilities,
- Reliable Internet connection,
- Longer mining/profit period due to host-based capabilities

Several instances of this type of cryptojacking malware have been found on cloud servers [Res, Hac, ARI, Unib, MV, clo]. In these attacks, attackers used different techniques to hijack the cloud server to inject cryptojacking malware. For example, in their 2020 annual report, Check Point Research [Res] observed that attackers integrate the cryptominer to the popular DDoS botnets such as KingMiner targeting Linux and Windows servers for side-profits. In another attacker instance [ARI], the researchers found an open directory containing malicious files. Further analysis revealed that the file contains a DDoS bot targeting open Docker daemon ports of Docket servers and ultimately installing and running the cryptojacking malware after the execution of its infection chain. In a similar attack instance [MV], the researchers noted a cryptojacking malware delivered using a CVE exploitation targeting WebLogic servers. Tesla-owned Amazon [Hac] and the clients of Azure Kubernetes clusters [clo] were exposed to cryptojacking attacks due to poorly configured cloud servers. Indeed, Jayasinghe et al. [JP20] showed that the count of cryptojacking malware targeting cloud-based infrastructure is increasing every year and affects more prominent domains such as enterprises.

IoT Botnet

IoT devices generally have small processing powers to perform basic tasks. It is being expected that there will be more than 21.5 billion IoT devices connected to the internet [Dep] by 2025. Attackers aim to create botnets with the collaboration of thousands of these IoT devices and perform several attacks such as DDoS due to their small processor, limited hardware, low-level security, and weak credentials, which was also exploited in the example of Mirai botnet's DDoS attack [McM]. Later, IBM researchers also found that the modified version of the Mirai Botnet also started to mine Bitcoin [MA]. Bartino et al. [BI17] states that there are several worms in IoT devices that hijacked them for mining purposes, and Ahmad et al. [ASKS19] proposes a lightweight IoT cryptojacking detection system to detect any cryptojacking attack that focuses on IoT devices.

Mobile

Cryptojacking malware samples targeting mobile devices inject cryptojacking script into their application and list the application in the application markets. Like every other type of cryptojacking attack, the mobile-based cryptojacking samples also have seen a great increase in the number of attacks. Because of this, both Google [goo] and Apple [Osb18] removed the cryptomining applications from their platforms. However, they still exist in alternative markets [DZG$^+$20]. The study by Dashevskyi et al. [DZG$^+$20] focuses on Android-based cryptojacking malware.

Moreover, mobile devices are generally not considered powerful enough for cryptocurrency mining because they generally use more restricted hardware and optimized operating systems (e.g., iOS and Android). Besides, the cryptocurrency mining process consumes extra battery and processing power, which may cause hardware problems such as overheating and apps to freeze or crash on mobile devices. Due to these reasons, cryptojacking attacks on mobile devices are not preferred by attackers, and they generally apply a mobile filtering method to opt-out mobile devices.

```
1 <script>
2       var miner=new CoinHive.Anonymous('Key', {
3          Threads:4, autoThreads:false, throttle:0.8);
4      if (!miner.isMobile()) &&!miner.didOptOut(14400)
5          { miner.start(); } }
6 </script>
```

Listing 2: The mobile device filtering method used in a cryptojacking sample.

Listing 2 is a recent cryptojacking sample with the mobile device filtering method found in a sample in our dataset. In line 4, the script automatically calls a mobile device detection function and starts the cryptocurrency mining process only if it is not a mobile device.

4.3.4 Target Cryptocurrencies

In this section, we give brief information about the most preferred cryptocurrencies by the attackers.

Monero

Monero has several advantages over other cryptocurrencies, making it favorable to attackers. First of all, Monero successfully implements and modifies the RandomX mining algorithm and CryptoNight hashing algorithm to prevent ASIC miners and give a competitive advantage to the CPU miners over GPUs via L3 caches [TNL18]. The Monero community aims to keep their network decentralized and allows even small miners to mine Monero. As in-browser cryptojacking malware can only access the CPUs of the personal computers through the browsers, Monero is ideal as a target cryptocurrency instead of other cryptocurrencies that are mined dominantly by other computationally more powerful ASIC and GPU miners. Second, Monero provides anonymity features through cryptographic ring signatures [BKM06, MSH+18], which makes the attackers untraceable. Thanks to these features of the Monero, attackers tend to mine Monero with their in-browser cryptojacking malware.

When we analyze the samples' cryptomining scripts in the PublicWWW dataset and their service providers' documentation, we found that all eleven service providers except Browsermine, CoinNebula, JSEcoin either use Monero or have the option to choose Monero in their scripts as a target cryptocurrency. This shows to 91% of the samples in the PublicWWW dataset use Monero to mine.

Bitcoin

In recent years, Bitcoin mining has seen enormous attention, which led to a dramatic increase in the difficulty target. ASIC and FPGA miners are the main reason behind this dramatic increase because the mining structure of the Bitcoin allows to build and use of specified mining hardware which is much more powerful and profitable than the CPUs and GPUs. The increase in difficulty target and disadvantages of CPU made the CPU mining infeasible and not profitable. Therefore, attackers who perform in-browser cryptojacking attack donot prefer Bitcoin mining. We also see that none of the service providers of the in-browser cryptojacking samples in our PublicWWW dataset supports Bitcoin mining. However, host-based cryptojacking malware can reach all the components of the victims' computer system and make Bitcoin mining on GPU and other high-performance computational resources of the computers. We also observe this in our VT dataset. We performed a keyword search for "bitcoin" on the AV labels of 20200 samples of both in-browser and host-based cryptojacking malware. We found that 7111 of 20200 samples are marked with a label containing the keyword "bitcoin". Even though this does not show that those samples are absolutely using bitcoin as a target cryptocurrency, but it is a potential indicator for the host-based cryptojacking samples mining Bitcoin based on the assumption of AV vendors are labeling the correct currency for the AV labels.

Other Cryptocurrencies

Cryptojacking is attractive for attackers as cryptomining can be parallelized among many victims. Therefore, it is possible for cryptocurrencies to allow distributed cryptomining. Both Monero and Bitcoin use PoW as a consensus method. However, instead of PoW, other cryptocurrencies utilize different consensus models such as Proof of Stake [Vas14], and Proof of Masternode [DSG14]. Most of these new consensus models do not depend on distributed power-based mining algorithms; therefore, cryptojacking is not an option for those currencies. For the cryptocurrencies that can be mined distributively [DHD$^+$20], the mining pools provide collective mining services to their participants. Other cryptocurrencies that are preferred by attackers are Bitcoin Cash [bit], Litecoin [Lit], and Ethereum [B$^+$14].

There are also several cryptocurrencies developed specifically for in-browser cryptomining activities. JSEcoin [JSE] is an example of them and offers also transparency. Other cryptocurrencies created for this purpose are BrowsermineCoin [Bro], Uplexa [Upl], Sumocoin [Sum], and Electroneum [Ele].

4.4 Conclusion

The rapid rise of cryptocurrencies incentivized the attackers to the lucrative blockchain and the Bitcoin ecosystem. With ready-to-use mining scripts offered easily by service providers (e.g., Coinhive [Coic], and CryptoLoot [crya]) and untraceable cryptocurrencies (e.g., Monero), cryptojacking malware has become an essential tool for hackers. The lack of mitigation techniques in the market led to many cryptojacking malware detection studies proposed in the literature. In this study, we first explained the cryptojacking malware types and how they work in a systematic fashion. Then, we presented the techniques used by cryptojacking malware based on the previous research papers, cryptojacking samples, and major attack instances. In particular, we presented sources of cryptojacking malware, infection methods, victim platform types, target cryptocurrencies, evasion, and obfuscation techniques used by cryptojacking malware. Moreover, we gave a detailed review of the existing detection and prevention studies as well as the cryptojacking analysis studies in the literature. Finally, we presented lessons learned, and we noted several promising new research directions. In doing so, this SoK study will facilitate not only a deep understanding of the emerging cryptojacking malware and the pertinent detection and prevention mechanisms but also a substantial additional research work needed to provide adequate mitigations in the community.

CHAPTER 5

IN BROWSER CRYPTOMINING FOR GOOD: AN UNTOLD STORY

5.1 Introduction

We summarize the main contributions of this chapter as follows:

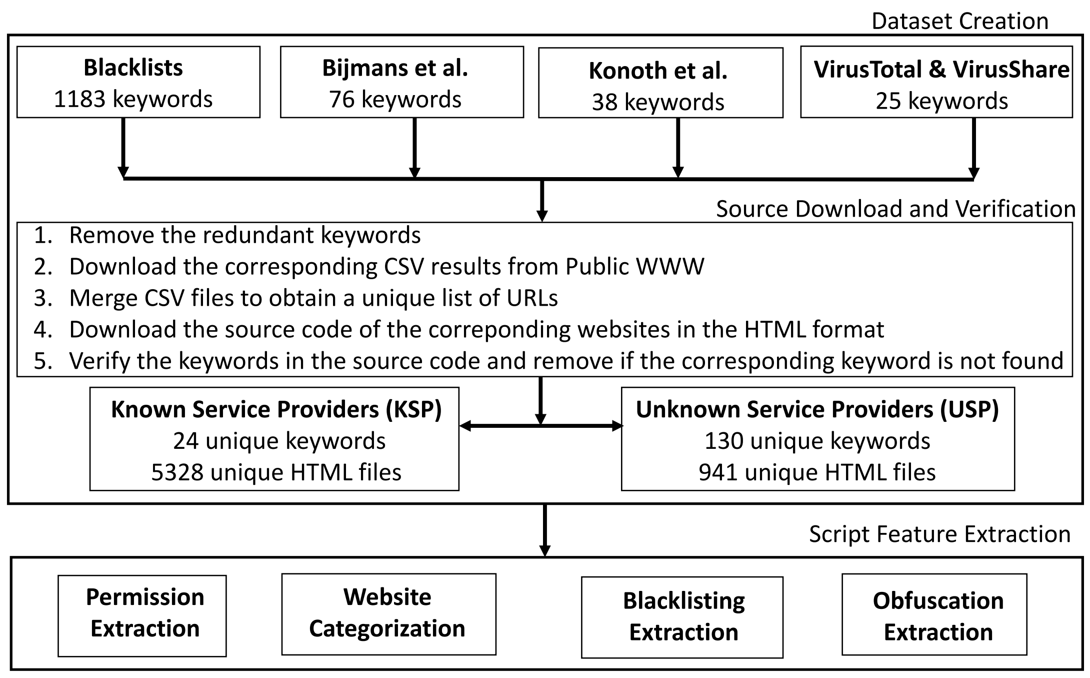
- We, for the first time in the literature, categorized in-browser cryptomining into two categories 1) Permissioned and 2) Permissionless (i.e., cryptojacking).
- We performed an empirical analysis with recent cryptomining samples[1] focusing on the permissioned cryptomining. For this, we collected a large number (i.e., **6269**) of unique cryptomining samples from **14** different service providers. We identified **24** unique keywords that can be used to detect the samples with those service providers. Moreover, we also identified **9** keywords for the consent detection and **4** obfuscated scripts.
- We perform profit, usability, and user consent analysis on the existing cryptomining scripts provided by the service providers found in our dataset.
- We proposed a novel consent evaluation framework for the service providers and presented our benchmarking results for the 14 service providers we detected in the dataset.

5.2 Dataset Creation & Methodology

In this section, we explain the methodology and tools we used for the dataset creation. The entire process is illustrated in Figure 5.1.

[1]In order to accelerate the research in this area, we also release our dataset and the list of keywords in the following link: `https://bit.ly/3sVj2cp`

Figure 5.1: Data collection process.


Dataset Creation
Blacklists
1183 keywords
Bijmans et al.
76 keywords
Konoth et al.
38 keywords
VirusTotal & VirusShare
25 keywords
Source Download and Verification
1. Remove the redundant keywords
2. Download the corresponding CSV results from Public WWW
3. Merge CSV files to obtain a unique list of URLs
4. Download the source code of the correponding websites in the HTML format
5. Verify the keywords in the source code and remove if the corresponding keyword is not found
Known Service Providers (KSP)
24 unique keywords
5328 unique HTML files
Unknown Service Providers (USP)
130 unique keywords
941 unique HTML files
Script Feature Extraction
Permission Extraction
Website Categorization
Blacklisting Extraction
Obfuscation Extraction

Keyword Collection: In order to find the cryptojacking samples, we used static keyword detection methods. Our primary source is the keyword blacklists released by browser extensions NoCoin [NoC] and MinerBlock [Mina]. We obtained a total of 1183 keywords from the merged blacklist of these two lists. Our second source for the keywords is the keyword lists released by other important studies Bijmans et al. [BBD19a] and Konoth et al. [KVM+18], in which we obtained 76 and 36 keywords, respectively. Our final keyword list is the keyword lists we found manually from the publicly known service providers. With this method, we extracted 25 keywords that could be used to detect cryptomining samples. After this process, we obtained 1322 a list of keywords.

Source Download and Verification: The collected list of 1322 keywords also includes duplicates and multiple keywords for the same service providers. Therefore, we removed the duplicates from the list and decided unique keywords for each known service provider. Then, we used PublicWWW [pub] to find the corresponding web-

Table 5.1: Service provider keywords.

Service Provider	**Keyword**
Authedmine	authedmine.min.js
	authedmine.eu/lib/1.js
	simple-ui.min.js
Browsermine	bmst.pw
Coinhave	cdn.minescripts.info
	coin-have.com
Coinhive	coinhive.min.js
	wp-monero-miner-using-coin-hive
	wp-monero-miner-pro
Coinimp	_client.start
CoinNebula	CoinNebula
Crypto-Loot	CRLT.Anonymous(
	cryptoloot.pro
DeepMiner	deepMiner.Anonymous
	deepMiner.Init
JSEcoin	load.jsecoin.com
Monerise	monerise_payment_address
Nerohut	nerohut.com/srv
Webmine	webmine.cz
Webminepool	WMP.Anonymous(
	wmp-site-key
	WMP.User
WebMinerPool	webmr.js
	webminer

sites containing those keywords in their HTML source code. PublicWWW is a web search engine tool, allowing us to search the HTML source of the websites, as of this writing, including over 500M websites. We downloaded the query result of all of the keywords. Some of the different keywords belong to the same URL; therefore, we again removed the duplicates. After removing the duplicates, we downloaded the corresponding source codes of the websites using a crawler. Our crawler checks the URL's HTTP response and connects to the webpage to fetch the HTML source code. After downloading the HTML source code, our crawler checks the related keywords from the keyword list and saves the HTML document to the related file. To avoid any discrepancies, we searched the keywords in the downloaded source code and removed samples that do not contain the keywords from the dataset. At the end of this process, we obtained a total of 6269 unique samples. We also obtained 24 unique keywords that can be used to identify 14 different service providers and corresponding 5328 HTML files as well as 130 unique keywords and 941 HTML source codes with unknown service providers. We labelled the samples with a known service provider as Known Service Provider (KSP) samples, and we used these samples for the rest of the analysis. In other words, we used 5328 unique samples with known service providers for all of the analysis in this study.

Table 5.1 shows the keywords we used for detecting the service providers. While deciding keywords, we tried to use the keywords that can not be changed easily. Instead of the source path, we used variables that are uniquely identifying the service providers.

Script Feature Extraction: In this step, our goal is to obtain more details about the usage of the cryptomining samples. Specifically, we are interested in the following features of the samples:

Table 5.2: Consent keywords.

Service Provider	**Consent Type**	**Keyword**
Authedmine	Permission	authedmine.min.js
Authedmine	Dashboard	simple-ui.min.js—
Coinimp	Notification	_client.addMiningNotification
Coinimp	Mandatory Mining	messageDiv
Crypto-loot	Dashboard	minui.js
JSEcoin	Permission	load.jsecoin.com
Webminepool	Dashboard	wmp-site-key
Webmine	UI	authedminer.js
WP Monero Miner[1]	Dashboard	wp-monero-miner

[1] WP Monero Miner is indeed not a service provider, however, it provides a plugin which can be used by multiple service provider and it has a dashboard style consent type defined in this study.

Table 5.3: Obfuscation keywords.

Service Provider	**Keyword**
Coinhive	authedmine.min.js
Crypto-loot	minui.js — crypta.js
Webmine Pool	base.js

- *User Consent Extraction:* In this part, we wanted to identify if the samples contain any user consent. For this purpose, we checked the documentation provided by the service providers. We obtained nine different keywords that could be used for user consent detection as well as the type of user consent. The list of keywords are given in Table 5.2. More details about different consent types are explained in Section 5.3.3.
- *Website Categorization:* In this part, our goal is to see if there is any correlation between the user consent and websites using the cryptomining script. For the website categorization, we used Webshrinker [weba], which provides a public web categorization service. However, it only returned a category for the half

of the dataset, where we manually labelled the rest of them following the same taxonomy of Webshrinker.

- *Blacklisting Extraction:* In addition, we wanted to identify if a website is blacklisted. For these, we used the public keywords lists released by the browser extensions by NoCoin [NoC] and MinerBlock [Mina], and if any keyword from the blacklists is detected in the source code of the sample, we labelled the sample as blacklisted.
- *Obfuscation Extraction:* Similar to the user consent extraction, we also observed that some scripts were obfuscated to avoid being blacklisted. We found 4 scripts given in Table 5.3 as obfuscated and labelled the samples utilizing these scripts as obfuscated.

5.3 Analysing the In-browser Cryptomining Ecosystem

5.3.1 Overall Analysis

In this section, we provide the overall distribution results of our dataset. Table 5.4 shows the list of 14 service providers we identified in our dataset and their related features while Table 5.5 shows the sample counts of those service providers.

The List of Service Providers. Using the service providers in [BBD19a, KVM+18] and extending them with publicly known service providers, we identified 14 service providers used by the samples in our dataset. We presented the list of service providers in our dataset in Table 5.4. We found that some of these service providers have discontinued their service. We used the Wayback Machine digital archive [webb] to access their documentation for those that are not active. We found that only 6 of 14 service providers are active as of this writing.

Table 5.4: The list of service providers we identified in our sample set and their other related features.

Service Provider	**Activeness**	**Permission Type**	**Currency**
Authedmine [Coid]	No	Permissioned	Monero
Browsermine [Bro]	Yes	Permissionless	BrowserMineCoin
Coinhave [Coie]	No	Permissionless	Monero
Coinhive [Coid]	No	Permissionless[1]	Monero
Coinimp [Coif]	Yes	Optional	Monero Mintme
CoinNebula [Coig]	No	Permissionless	No info
Crypto-Loot [Cryb]	Yes	Optional	Monero uPlexa
DeepMiner [Dee]	No	Permissionless	Monero Electroneum Sumokoin [Sum]
JSEcoin [JSE]	No	Permissioned	JSEcoin
Monerise [Monc]	No	Permissionless	Monero
Nerohut [Ner]	No	Permissionless	Cryptonight coins
Webmine [webd]	Yes	Optional	Monero
Webminepool [webf]	Yes	Optional	Monero
WebMinerPool [Webg]	Yes	Optional	Monero

[1] Coinhive is not providing a method to make the sample permissioned; however, it can be integrated to third party extensions to be made permissioned, which we marked them as permissioned sample.

We also marked the service providers according to their permission type. For the permission type, we marked the ones who are enforcing the consent method in the source script as "permissioned" because it does not give an option to remove the user consent to the website owners. On the other hand, we marked the ones which are not providing a method for user consent as "permissionless". Finally, we marked as both the service providers, which are providing a method for the user consent optionally. We found that among these 14 service providers, 7 of them are permissionless, while 5 of them are classified as "optional". Only Authedmine and JSEcoin embed the user consent in the script so that the website owner cannot remove it unless s/he modifies the original script.

We also extracted the cryptocurrency that can be mined based on the service provider documentation. As can be seen from Table 5.4, the majority of the service providers prefer Monero due to its anonymity features while some of them utilize their own cryptocurrencies such as BrowserMineCoin or JSEcoin.

Service Provider Distribution: Among all cryptomining service providers, the first and most popular one was Coinhive; however, Coinhive discontinued its operations as of March 2019. Since then, many other service providers have surfaced, and some of them still continue their operations. In our dataset, we noted that 43.5% of the samples (2380/5472) still have Coinhive script on their websites. If the script is not deployed locally or in a proxy, basically, these website owners are not making any money. Among the top five service providers with most samples, the only active service provider is Coinimp, with 1123 (20.5%) samples. Using the activeness information of the service providers, we found that in our dataset, a total of 1677 websites (30.6%) are using one of the active service providers. Moreover, we note

Table 5.5: Sample counts of 14 different service providers in our dataset.

Service Provider	**Total**	**Permissioned**	**Permissionless**	**Blacklisted**	**Obfuscated**
Coinhive	2380	152 (6.4%)	2228 93.6%)	2373 (99.7%)	34 (1.4%)
Coinimp	1123	56 (5%)	1067 (95%)	108 (9.6%)	13 (1.15%)
DeepMiner	492	0 (0%)	492 (100%)	16 (3.2%)	8 (1.6%)
JSEcoin	448	448 (100%)	0 (0%)	444 (99.1%)	53 (11.83%)
Authedmine	378	378 (100%)	0 (0%)	78 (20.63%)	224 (59.2%)
Crypto-Loot	210	6 (2.8%)	204 (97.1%)	181 (86.2%)	136 (64.7%)
Browsermine	155	0 (0%)	155 (100%)	8 (5.2%)	0 (0%)
Webmine Pool	84	5 (5.9%)	79 (94.0%)	7 (8.3%)	77 (91.7%)
WebMinerPool	83	45 (54.2%)	38 (45.8%)	83 (100%)	1 (1.2%)
Coinhave	58	0 (0%)	58 (100%)	35 (60.3%)	0 (0%)
Monerise	31	1[1](3.2%)	30 (96.8%)	0 (0%)	1 (3.2%)
Webmine	23	1 (4.3%)	22 (95.6%)	9 (39.1%)	0 (0%)
Nerohut	6	0 (0%)	6 (100%)	2 (33.3%)	0 (0%)
CoinNebula	1	0 (0%)	1 (100%)	1 (100%)	0 (0%)
Total	**5472**	**1092 (19.9%)**	**4380 (80%)**	**3345 (61.1%)**	**547 (10%)**

[1] This sample is using Coinimp and Monerise together and utilizing Coinimp's notification method.

that 144 samples[2] are using two service providers at the same time. Among these, in 139 samples, either of the service providers is Coinhive or Authedmine, while others contain Coinimp' scripts. This shows the popularity of these service providers as they are either used standalone or along with others.

User consent Distribution: As we showed in Table 5.4, seven service providers do not provide a method for the user consent while five of them provide an optional method, and Authedmine and JSEcoin are enforcing the website owner to ask for the user consent explicitly by embedding the user consent code snippet in the source of the script. Using the permission extraction method we explained in the previous section, we identified 1092 permissioned cryptomining scripts from all service providers.

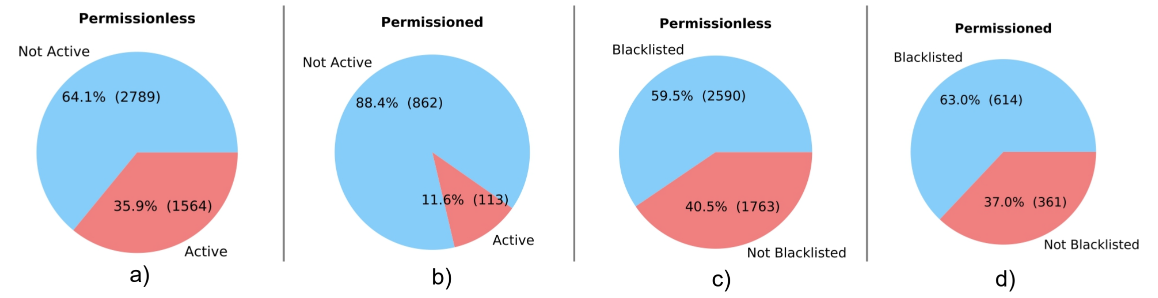
[2] Therefore, we further want to note that we have 5328 unique websites in our dataset, 144 of them are counted twice as they included two cryptomining scripts. We found no websites using more than two cryptomining scripts at the same time in our dataset.

Among all service providers, the three service providers with the most permissioned samples are JSEcoin, Authedmine, and Coinhive, where none of them are active as of now. We found 56 samples utilizing a permission method provided by Coinimp, which is active as of now. From its documentation, we found that the most basic script provided by Coinimp does not indeed utilize a permission method. Among service providers supporting both permissioned and permissionless cryptomining, a similar ratio is also observed for Crypto-Loot and Webmine Pool. For these, we conclude that most website owners are using the most basic script provided by the service provider.

Are all samples in the dataset blacklisted? As we explained in Section 5.2, using the keyword list publicly released by the browser extensions, we can decide if a given website is going to be blacklisted by one of these blockers. We found that these browser extensions blacklist 61.1% (3345/5472) of the websites in our dataset. Moreover, we also noticed that most of the samples utilizing Coinhive (99.7%) are blacklisted as it is the pioneer in the cryptomining business. Similarly, JSEcoin's blacklisting ratio is also very high as its parameters are embedded in the script, and the script is called in one line. In this method, when the source for the script is blacklisted, all of the samples using that same URL will be blocked. On the other hand, there are also service providers with very low blacklisting ratios such as Coninimp (9.6%), DeepMiner (3.2%), Browsermine (5.2%), Webmine Pool (8.3%), Monerise (0%).

Are the samples deploying obfuscated scripts? We noticed that some of the scripts are utilizing obfuscated scripts. We identified four such scripts, which is given in Table 5.3. In our dataset, we observed that 547 (10%) cryptomining scripts are utilizing one of these scripts, which may be suspicious. We also noticed that even though these scripts are obfuscated, they can be detected by the blacklists by the

Figure 5.2: Activeness distribution of a) permissioned (b) permissionless cryptomining samples. Blacklisted distribution of (c) permissioned (d) permissionless cryptomining samples.



code snippet calling the source of the script. We basically used this method in order to identify these samples. We also note that this method cannot detect the scripts located on a proxy server or locally with a different filename.

5.3.2 Comparison of the Permissioned and Permissionless Cryptomining

As shown in Section 5.3.1, we found that 1092 (19.9 %) samples in our dataset are utilizing a way to notify the user about the cryptomining operation that will be performed using the user's resources. In this section, our purpose is to compare the permissioned with permissionless samples to reveal the differences (if any) between them and explore the characteristics of the permissioned cryptomining. For this purpose, we perform a cross-correlation analysis among the features of the samples in our dataset. Particularly, we analyze the correlations between the following pairwise features: 1) activeness vs permission type, 2) blacklisting vs permission type, and 3) web category vs permission type. We present our results in the following subsections.

Activeness

As we noted in Section 5.3.1, not all of the service providers are active and continue their operations. For example, the most common permissioned cryptojacking service providers Coinhive/Authedmine, JSEcoin, WP Monero Miner do not continue their operations. Figure 5.2a and 5.2b show the activeness ratio of permissioned and permissionless cryptomining samples, respectively. We can see that the activeness ratio of the permissioned samples is 11.5%, while it is 35.9% for the permissionless cryptomining samples.

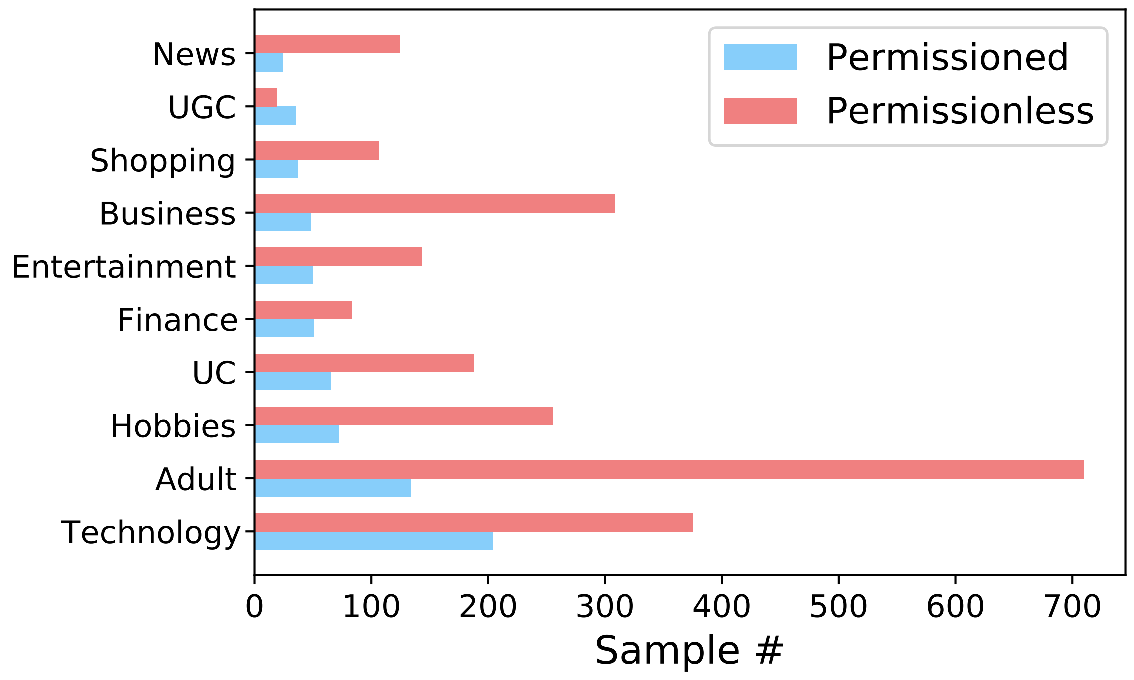
Blacklisting

Figure 5.2a and Figure 5.2b show the distribution of blacklisted websites for permissioned and permissionless cryptomining samples. We found that blacklisted websites' ratios are very close to each other (37.0% and 40.5%) for permissioned and permissionless. The reason for this is that blacklists do not really differentiate between the permissioned cryptojacking samples. Some service providers such as Authedmine or Webmine enforce the website owners to use a permissioned version of the scripts by embedding and sometimes making it impossible to modify by obfuscating to avoid being blacklisted. However, as we can see from the distribution, the blacklists do not consider if a script provides consent type to the website owner while choosing keywords for the blacklists. While this is to avoid the malicious cryptojacking samples, it also kills the responsible, legitimate website owners' functionality.

Web Category

Figure 5.3 shows the corresponding web categories used by the permissioned and permissionless cryptomining samples for those that can be categorized by Webshrinker. From Figure 5.3, adult content is the highest web category for permissionless samples, while the permissioned cryptomining samples are used on technology websites. On the other hand, overall distribution also shows the more normal distribution for permissioned samples, while the adult content dominates the permissionless samples.

Figure 5.3: Website categorization distribution of permissioned permissionless cryptomining samples. (UC: Under Construction, UGC: User-generated Content)



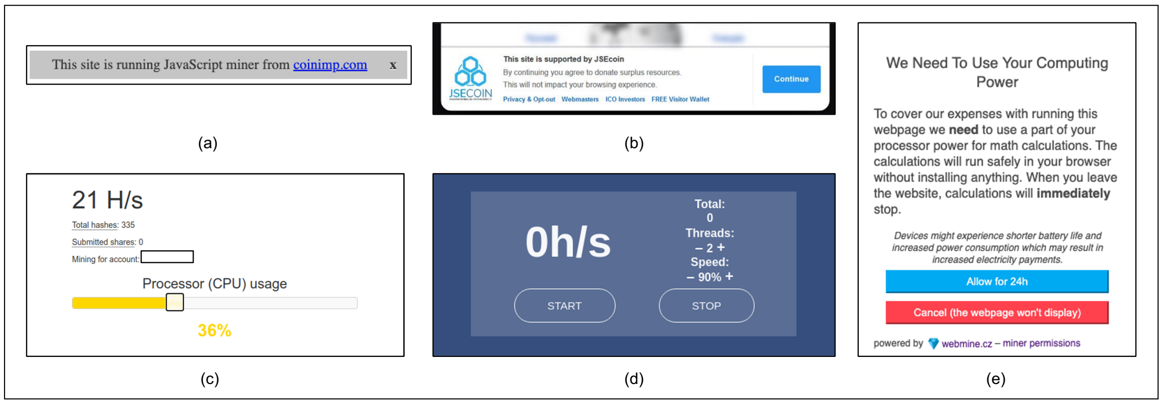
5.3.3 Extracting User Consent Methods

In Section 5.3.1 and 5.3.2, we analyzed the entire dataset and showed that the permissioned and permissionless samples mostly show similar behaviors, and they are mostly treated the same by the blacklists although the cryptomining business is shifting from the permissionless to permissioned cryptomining. In this section, we will focus on the permissioned cryptomining samples. Using the permissioned cryptomining samples, we identified five different types of consent types used by the samples: 1) permission, 2) dashboard, 3) notification, 4) mandatory mining, and 5) user interface (UI).

Table 5.6: The permission types provided by the service providers and their sample counts found in our dataset.

Permission Type	**Sample #**
Permission	666
Dashboard	255
Notification	51
Mandatory Mining	2
UI	2

Figure 5.4: (a) In-browser notification provided by CoinImp (b) The user view of JSECoin's permission consent type. (c) The user view of Webmine's UI consent type. (d) The user view of Webminepool's dashboard consent type. (e) The user view of Webmine's Mandatory mining consent type.



This site is running JavaScript miner from coinimp.com x

This site is supported by JSEcoin
By continuing you agree to donate surplus resources.
This will not impact your browsing experience.

We Need To Use Your Computing Power

To cover our expenses with running this webpage we **need** to use a part of your processor power for math calculations. The calculations will run safely in your browser without installing anything. When you leave the website, calculations will **immediately** stop.

Devices might experience shorter battery life and increased power consumption which may result in increased electricity payments.

Table 5.6 shows the permission types provided by the service providers and their sample counts found in our dataset. The results show that the permission, which we explain in the next subsection, is the most common consent type among the permissioned cryptomining samples in our dataset. The list of keywords, permission type, service providers offering these methods, and the keyword decision process are given in Table 5.2.

Figure 5.4 shows an example for each of the five consent methods, and in the next subsections, we explain these five consent methods in greater detail.

Notification

In the notification consent type, the user is notified by showing a pop-up screen. It only notifies the user without giving a selection to opt out. The Coinimp provides an example of the notification consent type, and it is shown in Figure 5.4a. The visibility of the notification can be adjusted by the website owner. For example, the text itself, text color, background color, as well as the size and position of the notification, can be configured. We note most of the samples in our dataset were using default settings; however, the notification can be even hidden from the user.

Moreover, this method is not mandatory and does not have to be placed in the code by the website owner. In this case, the mining will be starting automatically when the website is accessed by the user, which we call permissionless cryptomining. We found that only 51 (4.5%) of 1123 Coinimp samples are using this consent type in our dataset.

Permission

Permission consent type is similar to the notification, but it also gives the user an option to opt out from cryptomining. From the service providers in our dataset, we found that Authedmine and JSEcoin's scripts have a method to provide the permission consent type. Authedmine's script intentionally enforces this option in the source of the cryptomining script, and it does not provide an option to remove that part from the script to the website owner. In this way, the service provider's purpose is to avoid being blocked and to continue their service. However, as we have noted in Section 5.3.2, the blacklist does not consider being permissioned. On the other hand, as JSEcoin is originally proposed for transparency and accountability, similar to Authedmine's script, this part of the script is embedded into the source, and the website owner can not remove it. JSEcoin's permission example is shown in Figure

5.4b. As can be seen from the figure, the user can easily click the opt-out option and prefer not to allow the website to use his/her computational power.

Dashboard

Dashboard is another consent type where the user has the ability to start and stop the mining as well as configure the parameters such as the number of threads or CPU percentage. Configurable parameters may be essential for the websites who want their user to decide the use of his/her resources depending on the convenience. Webminepool provides an example of a dashboard consent types shown in Figure 5.4d. Some of the service providers implement dashboards so that mining starts automatically, but the users can set the parameters or stop the mining using the dashboard.

User Interface (UI)

UI is an improved version of the dashboard, where the user can set the parameters in an increased scaling as well as a better experience through the use of elements such as sliders. It requires more effort by the service provider, but it gives easy control to the users. We show an example of Webmine's UI consent method in Figure 5.4c. As can be seen from the figure, the user can see its resources used by the service provider and can also set the Processor CPU usage through the slider element.

Mandatory Mining

Mandatory mining is another consent type, in which if the user does not accept the mining, s/he will not be allowed to access to the website. The Coinimp service provider gives a method for this consent type. This method is also the rarest mining method in our dataset, and is found in only two samples. For example, Figure 5.4e shows the user view of Webmine's mandatory mining consent message.

5.4 Revisiting the Permissioned Cryptomining

In this section, our goal is to see if the existing service provider scripts can serve as a monetization tool, rather than an attack tool. For this purpose, we test them to see

- whether they can be an alternative revenue mechanism,
- whether they interrupt the end-users,
- whether they satisfy the consent requirements.

In the following subsections, we present our analysis results for each of them.

5.4.1 Can they be an alternative to advertisement?

The idea of using permissioned in-browser cryptomining as a monetization tool for the websites brings the question of how much profit they offer for the website visits. For this purpose, we deployed the scripts provided the active service providers found in our dataset on a sample website and calculated the total profit per user per minute. We run each experiment five times and calculated the average. Table 5.7 shows the real-time calculation of the monetary value for one regular user in one minute. As service providers may use different currencies, we converted them into the corresponding USD value as of this writing[3]. As can be seen from the table, Webminerpool offers significantly more profit compared to the other service providers.

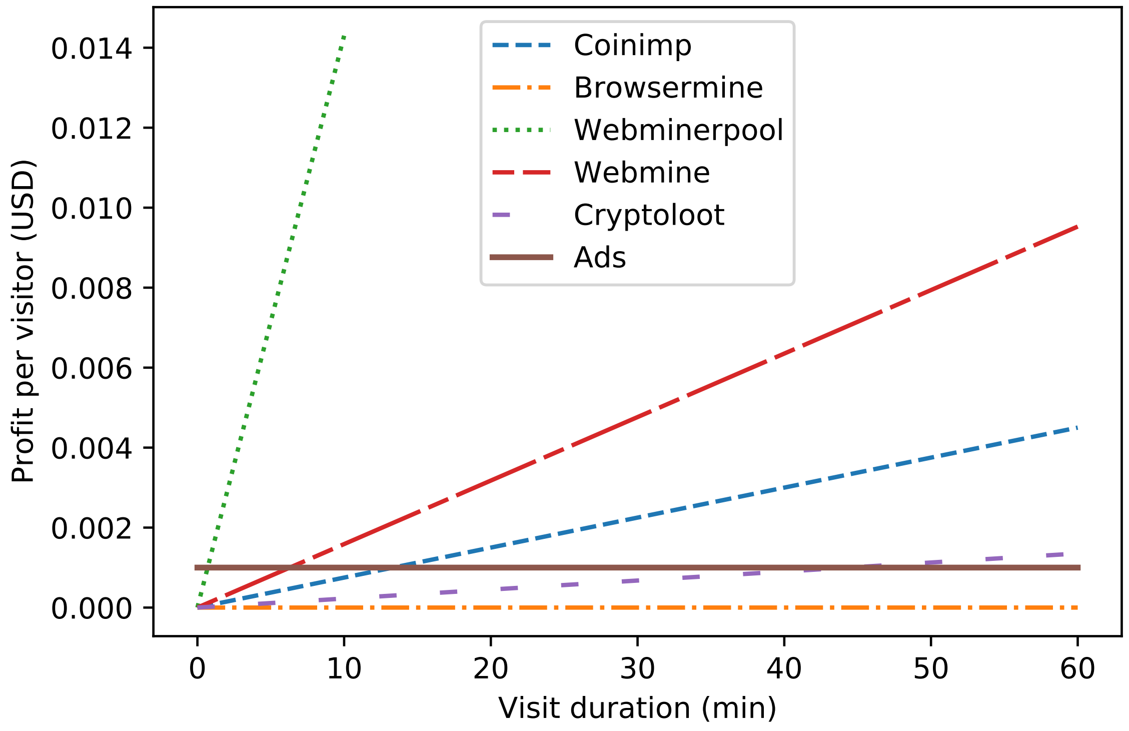
Moreover, even though an advertisement is assumed to pay a constant, the profit of a cryptominer increases linearly as the visit duration increases. Figure 5.5 shows the

[3]Nov 30, 2020.

Table 5.7: Real-time profitability analysis of active service providers.

Service Provider	Throttle	Time (min)	Currency	Profit (in currency)	Average Price (6 Months)	Profit (USD)
Coinimp	100	1	Mintme	0.03199108	0.0024 USD	0.000074
Browsermine	100	1	BMC	0.000000009101	0.047 USD	0.0000000004277
Webminepool	100	1	Monero	0.00001172	141.29 USD	**0.001655**
Webmine	100	1	Monero	0.000001312308	141.29 USD	0.000185
Crypto-Loot	100	1	Uplexa	0.17	0.00019 USD	0.0000323

Figure 5.5: Profit per user value for different visit duration.



profit per user value for varying visit duration. An average ads revenu is calculated per thousand impressions and on averages it varies between 0.5-2 USD [ads]. We assume 1 USD in our calculations. As we can see from the figure, while Webminepool can reach to the same amount of profit offered by the ads in less than a minute, Browsermine can not make the same amount of profit even for the 60 minutes of visit duration.

Table 5.8: Configurations used for evaluating the usability of in-browser cryptomining.

Conf 1	Chrome 2 tab static page
Conf 2	Conf 1 + 1080P Youtube Video
Conf 3	Conf 2 + Spotify
Conf 4	Conf 3 + Whatsapp App Video Download process
Conf 5	Conf 3 + Software uptader
Conf 6	Conf 4 + 4K video
Conf 8	Conf 4 + Virtual machine

5.4.2 Do they interrupt the users?

In this section, our purpose is to perform a usability test on the end users of in-browser cryptomining as greedy mining operations can be computationally challenging process for daily user computers. For the experiments, we used computes with four different CPUs: 1) Intel Core i5, 2) Intel Core i7, 3) Intel Xeon E5, and 4) Intel Xeon Gold for the throttle values 0.2, 0.5, and 0.8. All of the computers used in our experiments use the same RAM (16 gb, 1333MHz), similar SSDs and the same Linux distribution (Ubuntu 18.04 LTS). We run all experiments 5 times with the exactly the same applications to obtain more accurate results.

We present the configurations in Table 5.8. As a result of the analysis, we observed that it is possible to efficiently use in-browser mining scripts if the web page owner will not act greedy and keep the throttle rate under the 50%. Above 50% will dramatically affect the user's experience and user might tend to close the related tab due to the interruptions and overheating.

5.4.3 Do they satisfy consent requirements?

If implemented properly and responsibly, in-browser cryptomining can be used for good such as collecting donations as in UNICEF's case example [UNIa]. However, the lack of an evaluation framework for proper and responsible implementation of the permissioned cryptomining makes this technology's widespread adoption impractical, if not impossible, for the website owners. Therefore, here we first present a consent evaluation framework and then use it to rank the current service providers in our dataset.

A Consent Evaluation Framework

Herein, we suggest ten requirements, encompassing two categories: User Requirement (UR) and Webpage owner Requirement (WR).

User-related Requirements:

- *UR1: A method to notify the user to the website owner:* The webpage owners should at least inform the users about mining activity on the webpage. Some service providers offer built-in functions to inform users. This notification should be visible and explain in-browser mining employed by the webpage. The webpage owner may also place an informative link at the end of the notification for more information.
- *UR2: An option or link to the user to learn more info about the mining service:* With this method, webpage owners allow their users to learn more about the meaning of the in-browser mining and what they actually do during their visit to the related webpage. This option is also used by fundraising projects/webpages. These webpages aim to provide additional income for several civil society initiatives and humanitarian projects.
- *UR3: An opt-out screen to the user:* Webpages that operate in-browser mining should give options to their users. If the user does not want to give permission, the webpage should not force the user to allow the mining script. JSE coin, Webmine, and Authedmine let their users make this decision. Besides, webpage

owners may not want users to visit their webpage if they do not wish to opt out. This is a decent option for both users and webpage owners.

- *UR4: An option to stop the mining after starting during the session:* Some service providers have built-in functionality to stop and start mining operations manually. This feature allows the user to control the mining process and give the option to stop it if desired/needed. This method is very advantageous for both users and the webpage owners because if the user experiences interruption, they can stop the miner and continue surfing on the webpage.
- *UR5: An option to adjust the parameters (e.g., threads, CPU):* Most of the scripts in our dataset have standard CPU usage permissions (also known as throttle and threads variables) set by the website owners. Some websites and service providers allow their users to set how much they want to contribute to mining. Currently, this is the highest point of consent. If the user does not wish to contribute via mining, they can easily set the parameter to zero and continue surfing. Or if the website gets too greedy, which could interrupt the user, the user can limit the parameters according to his/her convenience.

Webpage Owner Requirements:

- *WR1: An option to change the notification message and its properties:* While some service providers let the website owner choose their own messages and change the properties (e.g., text and background color, location) of the notification, some others have mandatory features that directly come from the main library. This feature has several pros and cons for both users and the webpage owner. Firstly, when service providers let webpage owners arrange the parameters themselves, the webpage owner makes the notification message out of sight or very hard to notice. On the other hand, when service providers embed a mandatory mining script to the main library, the webpage owner must permit some third party to add content to the webpage.

- *WR2: An option to make the script mandatory:* Webpage owners may want to block users who do not exchange their computational power for webpage content. This option is beneficial for the webpage owner, and service providers should provide this option to the webpage owners.
- *WR3: An option to enable/disable the start/stop mining buttons:* Service providers generally make this option mandatory for the webpage owners, but this decision should be left to the webpage owner. As mentioned before, the webpage owner should choose what is going to be displayed on their webpage.
- *WR4: An option to change and modify dashboard location and design:* The dashboard and GUI-based control panels are user-friendly, but their design is generally not changeable by the webpage owners. Webpage owners should have the right to change the design of the dashboard under several restrictions.
- *WR5: A good documentation and guideline for website owners:* Service providers offer several functionalities to the webpage owners. While several service providers publish very poor documentation, some others provide high-quality documentation.

Table 5.9 shows how much the top service providers in our list fit into this framework. Some service providers fit into our framework with an impressive score (e.g., Webmine and Webminerpool). These service providers offer various options for both users and service providers; however, their user options are controlled by the webpage owners. If the webpage owner does not prefer to use any of them, service providers do not force them to do that. Besides, service providers like JSEcoin, Cryptoloot, and Authedmine force consent method usage, and this makes these service providers more user friendly.

Table 5.9: Summary of the service providers' consensual cryptomining features.

Service Provider	**UR1**	**UR2**	**UR3**	**UR4**	**UR5**	**WR1**	**WR2**	**WR3**	**WR4**	**WR5**	**Total (/10)**
Authedmine	◐	◐	●	◐	○	◐	◐	○	○	◐	4
Browsermine	○	○	◐	◐	◐	○	○	○	○	○	1.5
Coinhave	○	○	◐	◐	○	○	◐	○	○	◐	2
Coinhive	○	○	○	○	○	○	○	○	○	●	1
Coinimp	◐	◐	◐	○	○	●	●	○	○	●	4.5
CoinNebula	○	○	○	○	○	○	○	○	○	○	0
Crypto-Loot	○	○	●	●	●	○	○	○	○	●	4
DeepMiner	◐	○	○	◐	●	○	○	◐	●	○	3.5
JSEcoin	●	●	●	●	○	○	●	○	○	●	6
Monerise	○	○	○	○	○	○	○	○	○	○	0
Nerohut	○	○	○	○	○	○	○	○	○	○	0
Webmine	◐	○	●	●	●	●	●	○	●	●	**7.5**
Webminepool	○	○	◐	●	●	○	○	●	●	●	5.5
WebMinerPool	◐	◐	●	●	●	○	◐	◐	●	●	**7**

○: Does not satisfy ◐: Partially satisfies ●: Fully satisfies

5.5 Conclusion

In this chapter, we analyzed the characteristics of the in-browser mining ecosystem and the service providers' consent methods using a large dataset, which consists of 6269 unique websites containing cryptomining script in their source codes. We created the first consent focused in-browser cryptomining dataset in the literature and classified it under different consent methods. After the classification process, we analyzed our results and shared our findings. In light of the new classification process, we categorized consent types under different sections. We used the samples we found in the wild during these classifications. Another contribution of this research is a new evaluation framework for service providers and developers who want to implement a user consent-based in-browser cryptomining. This framework is adaptable and extensible for both academic research and service provider implementations. We believe this study will further increase the widespread adoption of legitimate cryptomining with user consent and knowledge and will increase the awareness on in-browser cryptomining.

CHAPTER 6

A LIGHTWEIGHT IOT CRYPTOJACKING DETECTION MECHANISM IN HETEROGENEOUS SMART HOME NETWORKS

6.1 Introduction

In this chapter, we performed extensive set of experiments to design and evaluate the best IoT cryptojacking detection mechanism. We first performed experiments to find the best-ranked features, the most accurate classifier, and the optimum training size. Then, we evaluated the effectiveness of our IoT cryptojacking detection mechanism with 12 novel experiments designed to assess various attacker behaviors and network settings. For this, we implemented the cryptojacking malware on IoT devices, a laptop, and a server in a safe setup. We explained several practical issues we came across during the implementation of cryptojacking on IoT devices in Section 6.5.3.

Summary of findings. In addition to our lightweight and highly accurate IoT detection mechanism, extensive experiments we performed to assess various attacker behaviors and network settings led to several interesting results worth nothing:

- We found that the highest malicious packet generation rate is still 72% less than the least packet generation rate of the benign dataset given in Table 6.3. This shows that the cryptojacking malware does not generate as many packets as daily web browsing and application data.
- We found that while in-browser cryptojacking malware uses evasion techniques such as limiting CPU and minimizing the network communication, host-based cryptojacking malware tries to take advantage of the victim's device at maximum computational power.
- We observed that an attacker targeting the server shows higher accuracy than other victim devices types (i.e., laptop and IoT), i.e., there is a higher chance

that the cryptojacking attacker will be detected during an attack targeting the server type device.

- We found that the malicious scenario with stealthy profit strategy (i.e., 10% throttle) is less accurate than robust (i.e., 50% throttle) and aggressive (i.e., 100% throttle) attack scenarios. This means that obfuscation methods of attackers can still create differences during the detection phase.
- We also found that a fully compromised scenario is the one most likely to be detected by our detection mechanism.

Contributions. We summarize the major contributions of this study as follows:

- We propose an accurate and efficient cryptojacking detection algorithm targeting IoT networks. Since we use network traffic-based features, our algorithm is capable of detecting both in-browser and host-based cryptojacking malware and can detect the cryptojacking malware with 99% accuracy with one-hour of training data without any dependence on cloud or on-device features.
- To evaluate our algorithm, we designed several novel experimental scenarios. We assessed both various attack configurations (i.e., cryptojacking types, profit strategies, victim devices, and throttle values) and network settings (e.g., fully or partially compromised). To the best of our knowledge, this study is the first to analyze various attack strategies and network settings in the area of cryptojacking detection.
- To overcome some of the practical issues during the implementation of cryptojacking malware in the IoT devices, we used novel techniques, which can be adapted by other studies in the future.
- In order to accelerate the research in this area, we released both the dataset and code.

Organization. The remainder of this part of thesis is organized as follows: In Section 6.3 defines the adversarial model and attack scenarios. Section 6.4 presents our modeling to convert the network traffic data into a binary classification problem. Then, in Section 6.5, we explain the details of our data creation process and perform initial data analysis on the raw dataset. Section 6.6 reports the novel design scenarios to evaluate our detection mechanism and the results. Section 6.7 discusses some of the challenges we came across during the implementation of the IoT cryptojacking malware and our novel techniques to overcome them. Finally, Section 6.8 concludes the study.

6.2 Background

6.2.1 Cryptocurrency Mining

Cryptocurrency mining is the process by which new cryptocurrencies enter circulation and is a critical component of the maintenance and continuity of the distributed blockchain ledger. The immutability of a blockchain network is provided by the consensus mechanism, which is cryptocurrency mining. Cryptocurrency mining is based on a puzzle based on the main features of cryptographic hash algorithms. These work-based consensus models are generally known as Proof of Work (PoW) consensus models.

Cryptocurrency mining is a laborious, costly process where one's reward depends on the luck factor. Work-based consensus mechanisms benefit from the diffusion feature of the hash algorithms to prevent miners from predicting hash values in a systematical pattern, and this feature also maintains the luck factor. However, due to the fact that miners are rewarded with cryptocurrencies for their work, it is an important source of income for many cryptocurrency investors. As the hardware investment increases, the difficulty, cost, and risk increase, while the chance of finding the block increases.

6.2.2 Cryptojacking Types

This section explains the details of different types of cryptojacking malware and their similarities and differences.

In-browser cryptojacking

The fast development of web technologies such as JavaScript (JS) programming language libraries and Web Assembly (WASM) open standards allow web developers to interact with the computer components via several instructions in the browser. The attackers also use these technologies to implement in-browser cryptojacking malware. For example, WASM provides the capability to run low-level instruction codes near-native speeds in browsers, and it is supported by all major internet browsers [was, NAB+21].

In-browser mining surged after the service providers (e.g., Coinhive) started to distribute easy-to-use cryptojacking scripts. With those scripts, the attackers can easily copy and paste an HTML script to the source of the webpage via several code injection vulnerabilities [Avi]. This piece of code creates a communication pipeline for the mining process and starts the mining process. In-browser cryptojacking scripts also include an identification number of the script owner. With this ID, service providers monitor the overall traffic coming from the specific script owner and make the reward distribution depends on this ID number. After the mining process started, the communication continues to receive the tasks and return the calculated results through the already established channel. The rewards are given to the account associated with the ID number periodically.

Host-based cryptojacking

Host-based cryptojacking malware aims to hide itself into the computer system of the victim's host and perform cryptocurrency mining. The main goal is placing the cryptojacking malware into the computational device of the victim as long as possible and keep it profitable.

The attackers distribute and locate their host-based cryptojacking malware with third-party applications [Ole,San], social engineering methods [McD], or using several vulnerabilities into victims' host system [MV]. Attackers also use IoT botnets [MA] to perform cryptojacking attacks. IoT devices have limited capabilities in terms of computational power. However, the incentive of the attackers is similar to the botnets here, in which the combined computational power of a large number of IoT devices can be used to perform a meaningful amount of cryptocurrency mining.

6.2.3 Machine Learning Tools

In this subsection, we explain the machine learning algorithms and methods we used during our experiments.

Feature Extraction and Selection Tools

Feature extraction is a dataset size reduction operation where an initial raw dataset is reduced to a more manageable and usable form for processing. Feature extraction methods aim to combine features with different property-based functions, effectively reducing the size of data that classifiers need to process and still describing the original dataset without any loss. There are several open-source tools that calculate thousands of different features automatically. In this study, we used *tsfresh* [tsfb] automatic feature calculation tool.

Feature selection is the next step of the feature extraction process. After the feature extraction tools calculate the features, the tools sort them in terms of their significance level [Lab68] (also known as P-value) and build relevance table [tsr]. With this method, we can easily eliminate the less significant features and improve our classification process.

Machine Learning Classifiers

Classification is the process where the algorithms categorize data into a given number of classes for the purpose of predicting the class of a given data feed. We used several different classification models (e.g., Logreg, KNN, SVM, RF) to train our models and receive the accuracy results in this study.

6.3 Adversary Model and Attack Scenarios

As mentioned before, IoT devices can be targeted by both in-browser and host-based cryptojacking malware. Attackers may also follow the path of different adversarial models and attack cases. We evaluated 7 Attack cases and made 12 discrete experiments to test the cryptojacking detection mechanism we proposed in this study. In this section, we explain how IoT devices are targeted by cryptojacking malware and how we track these adversaries in our experiments.

6.3.1 Cryptojacking with service providers

There are several different active service providers and thousands of web pages hosting cryptojacking malware on the Internet [TAU$^+$21]. Attackers generally take advantage of code injection vulnerabilities [Tre] of the webpages and web applications to inject their ready-to-use mining scripts provided by service providers.

In recent years, IoT devices gain new capabilities to provide better and more flexible frameworks to their users. These capabilities allow developers to integrate new technologies and run their code blocks via IoT frameworks. The attackers merge these framework capabilities with known vulnerabilities and abuse them to run their cryptojacking malware inside of these devices. We implemented WebOS IoT cryptojacking malware with LG's WebOS development framework [Webc] [1] and develop a basic WebOS application that calls cryptojacking script when the user starts running the application.

To be able to create a stable and controlled cryptojacking environment, we prepared a website under a controlled server and hosted several different *active* cryptojacking scripts. However, while some of the service providers can not provide a stable mining framework, we choose Webmine [webd] as our main service provider. We ran the script with combinations of different levels of computational hardware usage to observe the characteristic outcomes of those scripts.

6.3.2 Cryptojacking with Command and Control (C&C) servers

A C&C is a computer-controlled by the attacker to send commands to compromised devices. Attackers generally host these servers in cloud-based platforms for security and identity secrecy reasons. In the cryptojacking domain, C&C servers are working as a subset of a mining pool to receive and distribute mining tasks from the mining pool to compromised devices. Fig 6.1 represents a basic configuration of the C&C servers connected to the mining pools. The first well-known incident related to IoT botnets and cryptocurrency mining [Tre, MA] happened in 2017 under the famous Mirai IoT botnet [McM].

[1] `https://github.com/sokcryptojacking/detection`

Figure 6.1: Communication flow of C&C server-based cryptojacking from the compromised IoT devices to the servers.

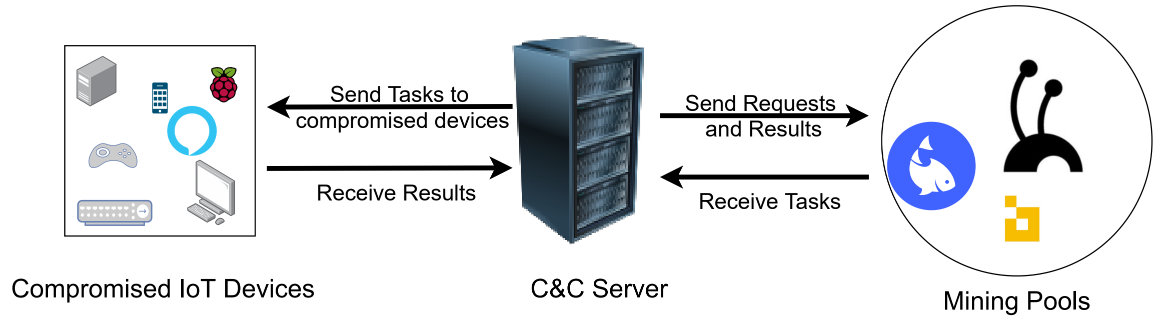
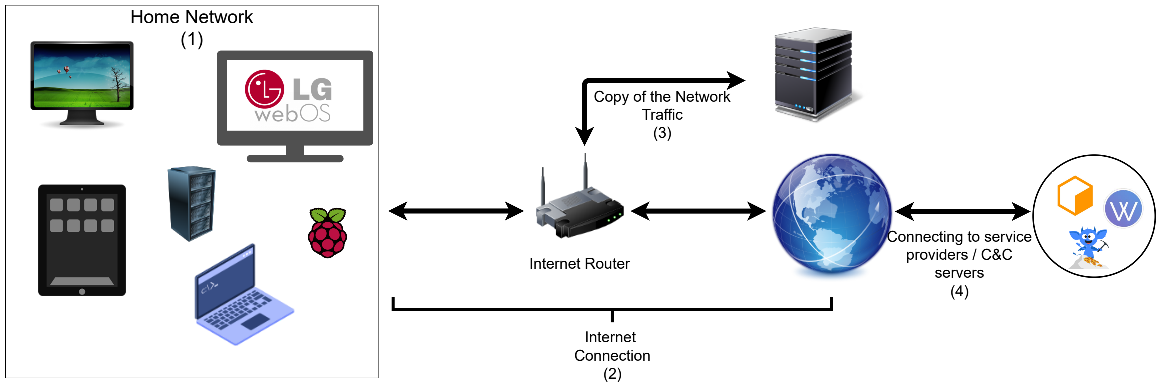


Figure 6.2: General flow of network data collection process.



Home Network (1)
LG webOS
Copy of the Network Traffic (3)
Internet Router
Connecting to service providers / C&C servers (4)
Internet Connection (2)

In this study, we focused on the communication pipeline between the compromised device and the C&C server. To demonstrate the process and data communication in this setup, we created a C&C server that sent mining tasks between different time periods. This time frequency can be changed depending on the block frequency of the blockchain network. In this work, we focused on Monero [Monb] and sent a mining task package within every two minute frequency. We successfully implemented this scenario with LG WebOS [Webc] Smart TV and other platforms we used for testing purposes.

6.4 IoT Cryptojacking Detection via Network Traffic

Network traffic classification and identification techniques have gained a lot of popularity in the last several years and it is a well-known technique to create user/device profiles on both server and local network sides [CLMS18, AFA+20]. In this study, we consider smart home network settings, where many IoT and non-IoT devices are connected to a router to be able to connect to the Internet. Each device can be identified via its MAC address. Therefore, we define devices in the network $(MAC_0, MAC_1, ..., MAC_n)$ for n in the network. We assume one or more devices in this network are compromised by the attacker to perform cryptocurrency mining on the behalf of the attacker and our purpose is to detect the devices performing by monitoring their network traffic for a certain time duration. For this, we use machine learning algorithms, which are trained with malicious and benign data beforehand. Devices generate continuous network traffic, which needs to be converted into a data format where the machine learning algorithm can predict whether the device is performing or not.

Before converting packets into proper format, we filter each packet using the following filter:

$$(MAC_{src} == MAC_i)\ OR\ (MAC_{dst} == MAC_i) \tag{6.1}$$

for a given device with MAC address of MAC_i. Then, we extracted following metadata from each packet:

$$Pkt_i = [MAC_i,\ timestamp,\ packet\ length] \tag{6.2}$$

At the end of this process, we obtain a series of packet lengths arrived at a given time for every device. Finally, we use a burst of 10 packets to calculate the features and we use these features to train/test the machine learning algorithm.

6.5 Dataset Collection

The data we focused on in our study is the network communication data between the IoT devices and cryptojacking service providers. In this section, we explain the main dataset collection and creation process by focusing on the topology, tools, methodology, and other implementation details we used for the IoT environment environment.

6.5.1 Topology

Figure 6.2 demonstrates our reference topology. We created a regular smart home network with several smart home devices, IoT devices, and personal computers (1). Data collection was performed under regular home-networking settings and all the devices performed unauthorized mining under a controlled environment. In our setup, all the devices in the home network are connected to the Internet via a single Internet router (2) similar to most home settings. There is also another computer in the network dedicated to collect all the Internet traffic with port mirroring [Por] and ARP rerouting/poisoning [NKK10] techniques (3). Finally, compromised devices in the network connect to the cryptojacking service providers or C&C servers of mining malware (4) to receive the tasks and return the calculated results. In this topology, our purpose is to be able to detect the compromised smart home devices that are performing unauthorized cryptomining inside the network.

6.5.2 Devices

We performed our experiments on four different devices representing varying computational power. Particularly, we performed the experiments on Raspberry Pi, Laptop, Tower Server, and LG Smart TV. Raspberry Pi and LG Smart TV represent the IoT devices in a real-life network while Laptop represents a regular device and Tower Server represents a computationally powerful device. Table 6.1 shows the devices we used during the experiments and their specifications.

Table 6.1: Device list used in the experiments.

Device	Representation	Hardware	Operating System
Raspberry Pi	IoT device	Cortex-A72 64-bit SoC 4GB RAM	Raspberry OS
LG Smart TV	IoT device	LG Quad Core Processor	WebOS 2.0
Laptop	Regular Device	Intel Core i7 9th Generation CPU 16 GB DDR4 RAM	Ubuntu 18.04 LTS
Tower Server	Powerful Device	Intel® Xeon® Gold 6314U Processor 192 GB DDR4 RAM	Ubuntu 20.04
Router	Internet Routing	Atheros QCA9563 Processor	OpenWRT_V.19.07.1

Moreover, we used TP-link Archer C7 V5 as our router in the given topology in Figure 6.2. This specific model allows us to configure port mirroring with its built-in features. We also used Ettercap [Ett] to manipulate the ARP protocol and forward the network traffic to the data collection computer's IP address. With this network configuration, we were able to collect all networking data with the Wireshark packet collector and analyzer [wir].

6.5.3 Implementation Methodology

The implementation of in-browser and host-based cryptojacking differs in several ways. In the next subsections, we explain the details of their implementations.

Implementing In-browser Cryptojacking

To be able to implement in-browser cryptojacking under a safe environment, we launched a basic WordPress [Wor] webpage which contains several different cryptojacking malware. We placed different HTML-based cryptojacking malware samples inside the source code of different pages of the test website for our purposes. After creating our experiment setup, we connected these pages with our test device and collected network traffic data for at least 12 hours for every use case scenario as explained in Section 6.3. We used the scripts distributed by Webmine.io [webd] and WebminePool [webf] service providers for the in-browser cryptocurrency mining.

Implementing Host-based Cryptojacking

Implementing host-based cryptojacking on Raspberry Pi and Server is straightforward. We downloaded the cryptocurrency mining binary MinerGate V1.7 [minb] and run it on our test device with our configurations. However, the implementation of host-based cryptojacking on the LG Smart TV was more challenging because the malware binary had to be placed on the victim's device and the victim's device had to allow to run the executable sample. We used the LG WebOS developer framework [Webc] to develop a basic application that runs cryptojacking malware as long as the application was running. In our scenario, we assumed that the malicious application could be an IP TV or streaming application. As a C&C server, we used a basic cloud server (With 1 GB RAM, 1 Core CPU, and Ubuntu Server 18.4 Operating System). After we created the malicious application that runs inside the WebOS-supported Smart TVs and the C&C server setup, we implemented two different models for the actual mining process as follows;

- **Without Connecting Mining Pool:** We made the first implementation with basic RandomX PoW algorithm [Monb,Ran]. When the application is activated, it sends a connection request to the C&C server, after that, the C&C server sends the mining tasks to the malicious application. The mining task contains three variables, the hash value, nonce value range, and the difficulty target. The RandomX implementation inside the malicious application starts mining inside the Smart TV until new command comes from the C&C server or finishes the

Table 6.3: Benign dataset sample sizes.

Dataset Name	Domain	Device	Currency	Total time (Minutes)	Packet Count	Packets per second (PPS)	Average_Package Size (Bytes) (APS)	Service Provider
Raspberry_Webmine.io_Robust	In-browser	Raspberry Pi 4	Monero	52	3519	1.13	**149**	Webmine.io
Raspberry_Webmine.io_Aggressive	In-browser	Raspberry Pi 4	Monero	735	12871	0.29	**163**	Webmine.io
Raspberry_WebminePool_Stealthy	In-browser	Raspberry Pi 4	Monero	521	9880	0.32	94	Webminerpool
Raspberry_WebminePool_Robust	In-browser	Raspberry Pi 4	Monero	527	6406	0.20	108	Webminerpool
Raspberry_WebminePool_Aggressive	In-browser	Raspberry Pi 4	Monero	1080	19643	0.30	113	Webminerpool
Server_WebminePool_Robust	In-browser	Server	Monero	382	16744	0.73	120	Webminerpool
Server_WebminePool_Aggressive	In-browser	Server	Monero	60	2539	0.71	101.17	Webminerpool
Desktop_WebminePool_Aggressive	In-browser	Desktop	Monero	720	234272	5.42	914	Webminerpool
Raspberry Binary	Host-based	Raspberry Pi 4	Monero	983	11745	0.20	95	MinerGate
Server Binary	Host-based	Server	Monero	1024	1198039	**19.50**	140	MinerGate
WebOS	Host-based	LG Smart TV	Monero	61	43173	**11.80**	117	AntMiningPool
Total				**6145**	**1558831**			

Dataset Name	Domain	Total time (Minutes)	Packet Count	Packets per second (PPS)	Average Packet Size (Bytes) (APS)
Bulk Data	Internet Data	18	2204727	2636.50	1114.5
Web Multiple	Internet Data	14.56	95388	91.78	567.25
Interactive	Internet Data	20.33	26144	355.97	249
Video	Internet Data	9.55	140009	243.33	956.3333333
Web Single	Internet Data	12.08	51381	71.46	638
Total		**74.52**	**2517649**		

nonce value range or finds the hash and nonce value that meet with the difficulty target.

- **Connecting Mining Pool via API:** The only difference between this implementation and the previous one is that the C&C server does not create mining tasks by itself, it receives them from the mining pool via its API framework and sends them over to the malicious application.

Both of the implementations we presented are just created for proof-of-concept purposes. For the real-world implementation, we used Ant mining pool [ant] for our dataset because it connects the real network behind the scene. And, we cover the data flow between the malicious application and the C&C server in our dataset. For the attackers who run botnet platforms (Botnet Admins), anonymity is always the priority. While running a self-mining environment requires having a fully synchronized full node and a lot of extra labor to maintain, it may also harm the anonymity of the attacker as well. For these reasons, we used the second option. Using the mining pool API is a lot more convenient and allows important extra flexibility to the attackers.

6.5.4 Labelling

While Wireshark was collecting all networking data, we ran all the attack scenarios we covered in Section 6.3 to collect the networking data. The network traffic generated by a device performing mining is labeled as malicious, while the dataset that is collected by a device that is not performing cryptocurrency mining is marked as benign.

6.5.5 Initial Data Analysis

We designed several different scenarios to collect malicious data with our controlled environment setup to assesses the different configurations that an attacker may use and different networks settings that are possible in real-life smart home environments. More details about the different configurations and our results are given in Section 6.6. On the other hand, we downloaded the benign dataset from a public repository [Ben]. The full details of the dataset are presented in Table 6.2 and 6.3.

Benign vs. Malicious

The main goal of this study is to be able to differentiate the malicious and non-malicious networking data from each other. For this purpose, we performed some initial data analysis on the raw data and list the outcomes as follows:

- **Packets per second (PPS) rate** is an important statistic to differentiate between the malicious and non-malicious data. As we can observe from Table 6.2, the highest PPS rate produced by the most powerful device we used while it was running binary cryptojacking malware. However, the highest malicious PPS rate is still 72% less than the least PPS rate of the Benign Dataset given in Table 6.3. This shows that the cryptojacking malware does not generate as many packets as daily web browsing and application data. This is an important challenge for both the data collection 6.2 and analysis phases. We discussed this challenge more in Section 6.7.

- **Average Packet Size (APS) rate** is the average size of all inbound and outbound network packets. It is a meaningful and relevant feature for purpose of the data discrimination process. APS rate creates almost the same pattern as the PPS rate. The highest malicious PPS rate is created by the Raspberry Pi 4 while performing in-browser mining with webmine.io [webd] service provider but the highest APS rate of the malicious data is still 35% less than the least APS rate of the benign data.

To sum up, network communication of cryptojacking malware and daily benign user shows very different characteristic features. In the rest of this study, we use this evaluated knowledge in the Overall Analysis (Section 6.6.1 for future selection process 6.6.1.

Host-based vs. In-browser Cryptojacking

The attackers mainly perform two different cryptojacking attacks to target different domains, in-browser and binary cryptojacking attacks [TAU+21]. To be able to see the different patterns generated by different devices under different attack scenarios, we performed in-browser and binary cryptojacking on all devices we used in this study and summarize the results in Table 6.2. We can summarize our observations as follows;

- **In-browser mining** always tends to generate a very small amount of PPS rate and APS rate on all devices.
- **For different service providers** of in-browser mining, there is no significant difference between the two service providers we used (i.e., Wembine.io and Webminerpool).

- **Binary mining** samples do not seem to use the obfuscation features used by the in-browser mining applications. They do not have any intonation to minimize their communication and keep themselves as stealth as possible.
- **For both In-browser and Binary** mining patterns, we can observe that in-browser mining always creates a very little amount of cryptojacking malware. There is no significant correlation between the hardware power, PPS rate, and APS rate. However, binary mining shows a completely different pattern where the APS and PPS rates are directly correlated with the power of the device.

In summary, there are important feature-based differences between binary and in-browser cryptojacking malware. We also use this evaluated knowledge in the Overall Analysis section 6.6.1 for future selection process 6.6.1.

Raspberry vs. Laptop vs. Server

Finally, we made the device-specific analysis to be able to see if there is any relevant feature that may allow us to differentiate the devices and understand which specific device category is infected by cryptojacking malware. We summarize our observations as follows,

- **All devices** give almost the same PPS and APS results for in-browser mining applications. We can deduce from the dataset outcomes, the service providers firstly receive the capabilities of the victims' host system and send mining tasks in terms of the power of the victims' host device.
- **For all devices** performing binary mining, we observe that the binary cryptojacking malware is correlated to the power of the victims' host system and both the PPS and APS rates are directly affected by the power of victims' host system as well.

To sum up, we found that while in-browser cryptojacking malware is trying to keep itself under a low profile and prevent high data density communication, host-based cryptojacking malware is not behaving in this way and generating huge network traffic. This is because of the method used by host-based cryptojacking malware, in which they are generally merged with other computationally heavy applications [TAU+21]. Therefore, the attackers are not worried about creating highly visible network communication.

6.6 Evaluation

In this section, we designed three sets of experiments to design and evaluate IoT cryptojacking detection mechanism that is accurate, efficient, and works in varying configurations and network settings.

- First, we design a set of experiments to design the optimum IoT detection mechanism with a highly accurate prediction rate and minimum training size and time.
- Second, to assess the success of the mechanism, we designed in the first part for different configurations such as different devices and throttle values.
- Third, we designed a set of experiments to assess the proposed mechanism in various smart home network settings.

We explain the details of these experiments and our results in the following subsections.

6.6.1 Designing an IoT Cryptojacking Detection Mechanism

After the dataset collection and labeling process, we created an overall dataset that contains the combination of all the malicious datasets and a benign dataset with an equivalent number of packets. Table 6.4 presents the dataset sizes and the total feature extraction and classification time of the overall dataset.

Table 6.4: The Overall Dataset Sample Sizes and total Feature Extraction & Classification Time

Dataset Name	**Dataset size**	**Total Classification Time**
Malicious	1557230	8.5 Hours
Benign	1556899	

In this sub-section, our goal is to design an IoT detection mechanism based on network traffic features and utilizing Machine Learning (ML) classifier for the classification. For this purpose, we performed the following steps:

- We use Feature Extraction to create feature vectors from the raw dataset.
- We select the best features and remove irrelevant features via a Feature Selection algorithm.
- We train and test several ML classifiers and decide which one performs best.
- We test the best algorithm with varying training sizes to optimize the training data and time as well as calculate the prediction time to assess the algorithm's feasibility in a real-world application.

We explain the details of these experiments and their results in the following subsections.

Feature Extraction

We used tsfresh [tsfb] to extract features from our dataset. The library tsfresh is a python package that automatically calculating statistical features from time-series data. In our case, we used ten packets for each feature vector and it created 788 different statistical features from its wide selection of statistical features [tsfa]. The features we used to train our overall dataset and other experiments are as follows:

- **Timestamp** is an important data extracted from every packet, and we used it to sort the given packets. While the mining process is running, data communication pattern changes depending on several variables. After we calculated the delta mean values of malicious and non-malicious traffic datasets, we observed an 11.4% difference between the mean values. This significant time difference leads us to use the data stamps of the network packages as a feature. We also noted that the devices mostly transmit stay-alive packets when no result is returned while they return application data packets with payload. And, while stay-alive packets are mostly transmitted periodically, the interarrival time between the packets decreases significantly while transferring application data packets.
- **Packet length** is another extracted data from every packet. All the miners must return some positive or negative result to the pool network. If the miner is directly connected to the mining network, they need to use several special protocols such as the stratum mining communication protocol used by Ethereum. With the IP addresses of the network connection endpoints, we labeled the data as malicious and non-malicious.

Features selection

Feature selection is the process of selecting a subset of relevant features for our models. With the feature selection process, we simplified and pruned our datasets to be able to smooth the dataset analysis and improve our results.

The relevancy scores of all the features in our dataset are not the same. To be able to optimize our extracted features and use only the most relevant features, we calculated the P-value. P-value (portability value) [Sch96] is a statistical model that calculates the probability of finding an observation under the assumption that a particular hypothesis is true. A smaller P-value (less than 0.05) is considered statistically significant. We found 290 statistically significant features for our datasets and train our models with those features. We used these 290 statistically significant features for the rest of the experiments in this study. We also share the extracted and relevant features in our dataset link.

Classifier selection

We implemented four Machine Learning classifiers to test the accuracy of the features we found in the previous subsection. During the implementation of these classifiers, we used the default parameters of scikit-learn [sci]. We used 75% of the data to train and 25% to test the classifier. We used Accuracy, Precision, Recall, F1 Score, and Test_roc as our metrics. The results of all classifiers are presented in Table 6.5.

Table 6.5: The Overall Dataset Classification Results

Classifier	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Test_roc**
Logreg	0.97	0.97	0.97	0.97	0.986
KNN	0.98	0.98	0.98	0.98	0.99
SVM	0.99	0.99	0.99	0.99	0.99
GNB	0.96	0.96	0.96	0.96	0.98

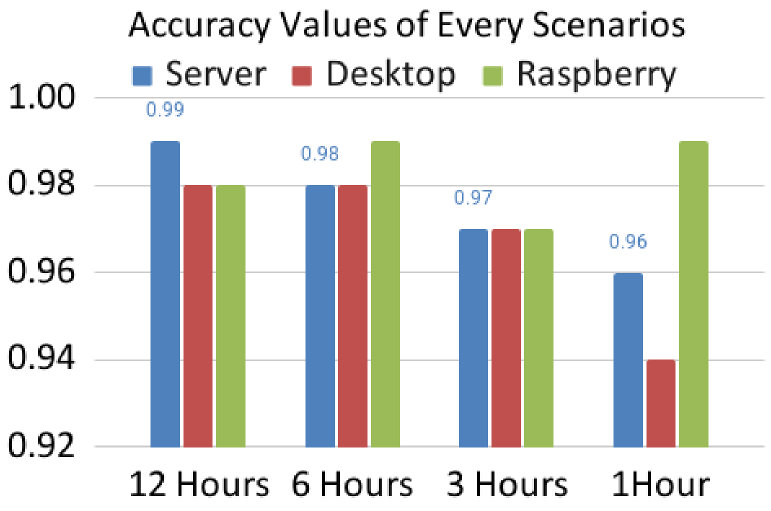
In this table, we used weighted average values calculated by scikit learn libraries.

Although all the classifiers provided really good results, our results showed that SVM performs the best among all the classifiers in terms of all metrics, aligning with many other detection studies in the literature [TAU+21]. SVM is a useful and well-designed classifier for supervised machine learning and it is very effective when there is a clear margin of separation. In addition, the SVM classifier is very stable, and small changes in the dataset do not cause important changes in results. Therefore, we decided to use SVM classifiers for our further analysis and implementation of other use case scenarios in the rest of the study.

Table 6.6: Dataset sizes of timing experiments

	Dataset Sample Sizes				
Dataset	**12 hours**		**6 Hours**	**3 Hours**	**1 Hour**
Server	*Malicious:*	838627	419313	209656	69885
	Benign:	837701	418850	209425	69808
Desktop	*Malicious:*	234272	117136	58568	19522
	Benign:	234448	117224	58612	19537
Raspberry	*Malicious:*	7829	3914	1957	978
	Benign:	8265	4132	2066	1033

Figure 6.3: The accuracy for every one feature vector during classification (in seconds).



Training Size and Timing Results

In this section, we performed experiments with varying training sizes. With this experiment, we analyzed the effects of the dataset collection time on classification accuracy and overall classification time. To obtain a reference result, we firstly fit the times of representative datasets to 12 hours by decreasing the size of the original dataset and fit them into 12 hours, 6 hours, 3 hours, and finally 1 hour and repeat the classification to measure accuracy and time-based values for each training size. Figure 6.3-6.6 summarizes 4 different results we cover under this section.

Figure 6.4: Prediction times for every one feature vector during classification (in seconds).

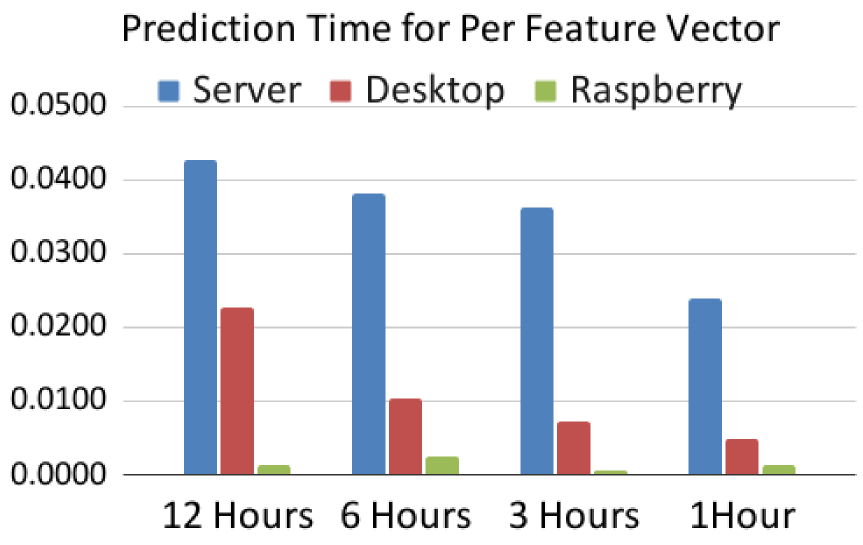


Figure 6.5: Classification time used for training (in minutes).

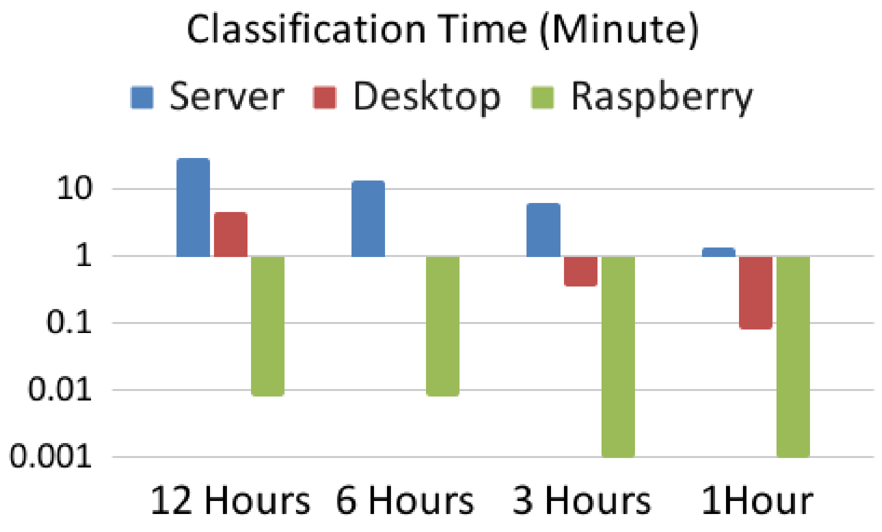
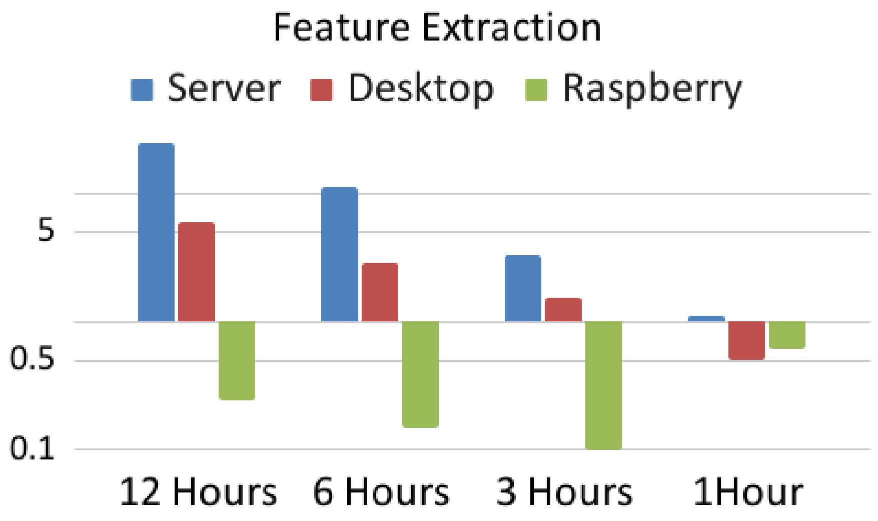


Figure 6.6: The time used for feature extraction (in minutes) for each training size (i.e., 1, 3, 6, and 12 hours).



- **Accuracy** is the first metric we checked after we rerun the classification for every dataset. As can be seen from the results in Figure 6.3, the accuracy did not change dramatically, and even when we only used less than 10% of the original dataset such that we did not receive any result below 94%. In addition to this, our model achieved to detect in-browser cryptojacking with 99% accuracy with only one-hour long data collection. It shows that the model we used is not extremely dependent on the dataset size, and it can give accurate results even with a shorter data collection duration.
- **Prediction time for per feature vector** represents how much time we need to predict the class per feature vector. This is a realistic metric we calculated to see how long it would take to predict the result of the collected dataset on a regular machine. After we evaluate our experiments, we saw that the time needed to train per feature vector is related to the size of the dataset. For bigger datasets, it takes more time to evaluate per data. However, after the feature

extraction process, we receive successful optimization results as low as 100-150 ms for each vector.

- **Feature extraction and classification** time represents the needed time for calculating features and classify these features for each dataset. As can be seen from Figure 6.5 and 6.6, the total time needed for the feature extraction and classification is directly correlated to the dataset size. In addition to this, with the very low feature extraction and classification time results, we still managed to get near-perfect results.

Overall, the results show that we achieved to implement a successful detection system without causing a lot of overhead on the devices or inside the network. We can also conclude that we can use slightly smaller datasets to train our model without sacrificing our dataset's accuracy level and trust factor.

6.6.2 Evaluation With Different Adversarial Behaviours

In this sub-section, our goal is to assess the IoT cryptojacking detection mechanism we designed in Section 6.6.1 with various attack configurations. An attacker can target different victim devices, pursue different profit strategies or have a choice of cryptojacking type of either in-browser or host-based. We have extensively evaluated our mechanism by performing a comprehensive set of experiments to test these three configurations. All the scenarios and experiments are implemented under the same 290 features we extracted during the experiments of Section 6.6.1. This implementation methodology allows us to observe the results of how effective to use one feature set for different use case scenarios. In the rest of this section, we summarized the detailed experiment results for each scenario.

Attack Case 1: Server vs. Desktop vs. IoT

In this scenario, we set up our environment with different kinds of devices. Our goal is to see if there are any differences between the detection accuracy of the cryptojacking malware running on each device. For this purpose, we first created a balanced dataset for this scenario, in which we created a balanced dataset for each device. Table 6.7 lists the dataset sizes and the devices we used to implement this scenario.

Table 6.7: Dataset sizes of Scenario 1

Malicious Dataset Device	**Benign Dataset**	**Malicious Dataset**
Server	1219381	1217322
Desktop	234448	234272
IoT	102506	102117

Attack Case 2: Profit Strategies

Throttle adjustment is one of the major obfuscation methods used by the attackers [TAU+21] and almost all of the active and inactive service providers provide this option to their clients. With the throttle adjustment feature, attackers can set the total hardware usage on victims' hots devices It makes this feature an important use case for our detection mechanism. While increasing the throttle value increases the profit, it also increases the likelihood of being detected by the system. Our goal with this experiment is to see if changing the throttle value has any impact on the detection accuracy.

- **Aggressive** cryptojacking malware focuses on making a maximum profit in minimum time. They are generally seen on the websites visited for a short time (less than 2 minutes) such as e-commerce, dictionary, and legal/illegal data downloading websites. The aggressive cryptojacking malware parallels the mining task to use all the remaining CPU power; therefore, it deteriorates the user experience dramatically. Thus, the malware detection algorithms or even the user can easily notice and detect the cryptojacking malware. We set the throttle value to 100% to observe such an attacker behavior.

- **Robust** cryptojacking malware employs multiple scripts to keep working even if the primary service provider fails. Service provider failure can occur not only on the server-side, but also on the victim side. For example, some extensions may block the connection to a service provider. In this case, the script can continue mining with other service providers via this strategy. We set the throttle value to 50% to observe such an attacker behavior as it only runs one of the scripts at a time.
- **Stealthy** cryptojacking malware aims not to be noticed by the user and by threshold-based detection algorithms deployed on the user side. It utilizes CPU limiting for long-term profit and is widely seen on illegal media/content websites (e.g., illegal film/series streamers, reading content pages, forums), where the users tend to spend a relatively longer time. It acts as an active content provider by using the same amount of CPU power as online flash games and media streamers. Therefore, it is hard to detect. We set the throttle value to 10% to observe such an attacker behavior.

Table 6.8: Dataset sizes of Scenario 2

Dataset Scenario	**Benign Dataset**	**Malicious Dataset**
Throttle (Aggressive)	1293096	1293801
Throttle (Robust)	35265	36021
Throttle (Stealthy)	9877	9880

Attack Case 3: In-Browser vs Binary

Host-based and in-browser cryptojacking are the two main application areas of the cryptojacking malware and to be able to claim that we can detect both in-browser and host-based cryptojacking malware, we must also have the ability to apart them from each other. In this scenario, we tested these two main attack surfaces of cryptojacking malware. Our goal in this experiment is to see if our detection system can successfully classify both the in-browser and the host-based cryptojacking malware with the same set of features. We used two balanced datasets for this experiment as can be seen from Table 6.9.

Table 6.9: Dataset sizes of Scenario 3

Cryptojacking Type	Benign Dataset	Malicious Dataset
In-Browser	290181	289130
Binary	1256659	1258894

Results

Table 6.10 presents the accuracy results of all three different attacker cases. In all three scenarios, we observed some differences between the different configurations.

Table 6.10: Classification results of all scenarios

	Attack Case	Accuracy	Precision	Recall	F1 Score	Test_roc
Scenario 1	Server	0.99	0.99	0.99	0.99	0.99
	Desktop	0.98	0.98	0.98	0.98	0.99
	IoT	0.98	0.98	0.98	0.98	0.98
Scenario 2	Aggressive	0.98	0.98	0.98	0.98	0.99
	Robust	0.98	0.98	0.98	0.98	0.98
	Stealthy	0.97	0.97	0.97	0.97	0.98
Scenario 3	In-Browser	0.97	0.97	0.97	0.97	0.99
	Host-Based	0.98	0.98	0.98	0.98	0.99

In this table, we used weighted average values calculated by scikit-learn libraries.

In Scenario 1, we successfully received an almost perfect score from all three experiments we made. However, the server shows a higher accuracy among all victim devices types, i.e., there is a higher chance that the cryptojacking attacker will be detected during an attack targeting the server type device.

In Scenario 2, the malicious scenario with stealthy profit strategy (i.e., 10% throttle) is less accurate than robust (i.e., 50% throttle) and aggressive (i.e., 100% throttle) attack scenarios. As we mentioned in Section 6.3, attackers use these obfuscation methods to keep their miners safe from the detection methods. While 97% accuracy value can still be considered almost perfect, it also means that obfuscation methods of attackers can still create differences during the detection phase.

Finally, in scenario 3, we can see the effect of the combination of obfuscation methods on in-browser cryptojacking detection samples. The general results for in-browser cryptojacking malware are just one step behind the host-based cryptojacking samples. However, if we have a look at the overall results of SVM in Table 6.5, the combination of in-browser and host-based cryptojacking malware leads us to 99% accuracy rates.

While cryptojacking malware has the ability to infect different devices, the proposed malware detection system needs to be able to detect the ongoing cryptojacking process without any device dependency. We saw that our extracted features could achieve near-perfect scores without any device dependency from the results of three scenarios and eight discrete experiments.

6.6.3 Adversarial Models of Compromised Device Numbers in Smart Home Network

In the previous section, Section 6.6.1, we successfully implemented discrete scenarios for 3 different datasets. In this section, we investigate different adversarial models implemented inside a simulated network that can be seen in smart home environments.

We implemented four different scenarios and presented their results in the rest of this section.

Scenario 1 - Fully compromised

In this scenario, attacker(s) exploit all devices in the home environment. This scenario could apply to several network-based attacks. For this experiment, we used our overall dataset (i.e., all malicious data we collected). While it is an extensive dataset with big-size network packages, it took a long time to extract features and make the classification process with the SVM classifier. We give the dataset sizes and total time spent for feature extraction and classification processes in Table 6.11.

Table 6.11: Dataset sizes and classification times of adversarial model analysis scenarios

	Test Case	**Malicious Dataset Size**	**Benign Dataset Size**	**Feature Extraction Time**	**Classification Time**
Scenario 1	Fully compromised (All)	1556899	1557230	51	215
Scenario 2	Partially compromised (IoT + Laptop)	246017	252647	3.7	7.5
Scenario 3	Single compromised (IoT)	12398	11745	0.3	0.02
Scenario 4	IoT compromised (IoT + IoT)	275951	275844	4.27	6.41

Scenario 2 - Partially compromised

Scenario 2 presents two different devices from different categories exploited by the attacker(s) with different cryptojacking attacks. While the IoT device was exploited by host-based cryptojacking and performed binary mining operation, the laptop device was compromised with an in-browser cryptojacking attack. In this scenario, most probably, one needs to consider two discrete attacks performed by different malicious entities. Still, in this scenario, both devices are using the same gateway (e.g., router, ADSL modem, Ethernet port) for internet communication.

Scenario 3 - Single compromised

When only one device in the network is compromised by the attackers, it makes it harder to detect malicious device. This scenario specifically holds a higher level of importance because, inside a smart home network environment, there can be a number of different IoT devices that may exist. In addition, while attackers use specific vulnerabilities to inject their malware, there can be only one or very few devices that may be exploited by that specific vulnerabilities.

Scenario 4 - IoT compromised

Our last scenarios inspired by Scenario 3. In this scenario, we discussed, what if two IoT devices from different domains were exploited by 2 different kinds of cryptojacking malware. To be able to simulate this environment we used LG WebOS device exploited by malicious application hosts host-based cryptojacking malware and Raspberry Pi 4 that exploited by the malicious webpage to perform in-browser mining. We received a near-perfect score for our last experiment as well.

Results

In this section, we designed and implemented several scenarios to track a real home environment. In the first scenario, all of the devices are compromised. In the second scenario, we used two different types of compromised devices that were exploited with different types of cryptojacking malware. In the third scenario, only one device inside the home network performing a very limited amount of mining was compromised. Finally, in the fourth scenario, we tried the same scenario with two different IoT devices (Raspberry Pi and WebOS Smart TV). The results of these two scenarios have importance because these two scenarios reflecting most of the Mirai [McM] and other known IoT botnet [BI17] attack scenarios. Our result shows that the fully compromised scenario is the one most likely to be detected by our detection mechanism, but we also see that all of the scenarios show a near-perfect accuracy. This implies that the detection model and feature set we implemented can successfully detect various home environment attack scenarios.

Overall, the combination of our selected classifier and feature set successfully detect the cryptojacking malware with high accuracy without being affected by any of the known attempts and obfuscation that may have been used by the attackers.

Table 6.12: Results of network settings analysis scenarios

	Test Case	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Test_ROC**
Scenario 1	Fully compromised (Overall)	0.99	0.99	0.99	0.99	0.99
Scenario 2	Partially compromised (IoT + Laptop)	0.98	0.98	0.98	0.98	0.99
Scenario 3	Single compromised (IoT)	0.98	0.98	0.98	0.98	0.99
Scenario 4	IoT compromised (IoT + IoT)	0.98	0.98	0.98	0.98	0.99

6.7 Discussion

Challenges of creating an IoT cryptojacking dataset

We found that the traffic between the mining server and the client does not create heavy network data. Cryptojacking service providers are specially designed not to create as much communication as regular Internet communication and mining processes. Table 6.2 and 6.3 shows the difference between the packet per minute of cryptojacking malware and a regular Internet connection. In addition to this, after we evaluated our tests on the service providers we used [webd, webf], we saw that average service providers use different IP addresses during the mining operation to communicate with the victims' client device and deliver the mining task via encrypted application data. This behavior of in-browser cryptojacking malware makes them harder to track inside busy networks. Moreover, it is also an indicator of the impracticability of IP and hosting-based static blacklisting implementations. To be able to monitor the nature of the in-browser cryptojacking, one neededs to create a special environment as we explained in Section 6.2.

Another important challenge we faced during our data collecting process is the asymmetric communication between the client and malicious IP addresses. In a normal data communication process over the Internet, the data packets create some symmetry between the client and the server but in the cryptojacking samples, there is no symmetry or pattern between the client and server. 88% of communication packets were produced by the client during the mining process and only 12% of packets were created by malicious servers. Furthermore, as mentioned above, this 12% portion was created by different IP addresses, when we summarize our findings, every malicious server creates only 0.89% of the malicious communication. It makes it harder to analyze the collected data and keep track of the communication pipelines between the client and the malicious server.

6.8 Conclusion

This chapter proposes an accurate and efficient cryptojacking detection mechanism based on the features extracted from network traffic. Our mechanism is able to detect both in-browser and host-based cryptojacking malware. We achieved 99% detection accuracy with one-hour network traffic data used to train the machine learning classifier. We also analyzed 12 different novel attacker behavior to test our mechanism in attack configurations and home network settings. In addition to, we also simulated cryptojacking attacks to several different platforms to see the efficiency of our detection mechanism. We show that different configurations that the attacker may use and different network settings that the mining is performed on affects the detection accuracy. Moreover, we share the network traffic we collected and the code publicly to accelerate the research in this area.

CHAPTER 7

CONCLUDING REMARKS AND FUTURE WORK

In this thesis, we performed a novel, empirical analysis of in-browser cryptomining processes, while primarily focusing on authorized cryptomining. First, we present a systematic overview of the current state of cryptojacking malware using the information obtained from previous research and two large datasets of samples and over 40 attack instances. We then performed an in-depth, first of its kind analysis on permissioned cryptomining using a dataset of 6269 unique cryptomining scripts. Our evaluation and analysis concludes that permissioned in-browser cryptomining has the potential for widespread adoption and could be a legitimate monetization tool if implemented responsibly.

There are two types of Bitcoin- and blockchain-related malware seen in the wild: those that use the Bitcoin and blockchain infrastructure to exploit the victim; or those that use the traditional malware attacks such as key stealing, social engineering, or fake application attacks to exploit Bitcoin and blockchain users. Cryptojacking attacks use the Bitcoin and blockchain infrastructure to exploit the victim's computational power; however, Bitcoin and blockchain users are also exposed to many traditional malware attacks. These attacks specifically aim to obtain Bitcoin and blockchain users' private keys through social engineering methods [Red, Phi, Bod], fake wallets [Loo, Stea], and key-stealing trojan malware [Para, Cima, Steb]. Although these attacks and their countermeasures [AAUA21, Aea19, Cea19] have been studied extensively in the literature [HL15], their impact in the Bitcoin and blockchain domain has not been investigated yet and can lead to new research directions.

Finally, in this thesis, we analyzed the characteristics of the in-browser mining ecosystem and the service providers' consent methods using a large dataset, which consists of 6269 unique websites containing cryptomining script in their source codes.

We created the first consent focused in-browser cryptomining dataset in the literature and classified it under different consent methods. After the classification process, we analyzed our results and shared our findings. In light of the new classification process, we categorized consent types under different sections. We used the samples we found in the wild during these classifications. Another contribution of this research is a new evaluation framework for service providers and developers who want to implement a user consent-based in-browser cryptomining. This framework is adaptable and extensible for both academic research and service provider implementations. We believe this thesis will further increase the widespread adoption of legitimate cryptomining with user consent and knowledge and will increase the awareness on in-browser cryptomining.

[AAUA21] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Kemal Akkaya. A usable and robust continuous authentication framework using wearables. *IEEE Transactions on Mobile Computing*, 20(6):2140–2153, 2021.

[ACA+17] Abbas Acar, Z Berkay Celik, Hidayet Aksu, A Selcuk Uluagac, and Patrick McDaniel. Achieving secure and differentially private computations in multiparty settings. In *Privacy-Aware Computing (PAC), 2017 IEEE Symposium on*, pages 49–59. IEEE, 2017.

[ads] Adsense cpm rates in usa: 2020. https://adcpmrates.com/2020/08/16/adsense-cpm-rates-in-usa/. Accessed: 2021-4-7.

[Aea19] Abbas Acar and et al. A privacy-preserving multifactor authentication system. *Security and Privacy*, 2(5):e88, 2019.

[AFA+20] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2020.

[ALB+19] Abbas Acar, Wenyi Liu, Raheem Bayeh, Kemal Akkaya, and Arif Selcuk Uluagac. A privacy-preserving multifactor authentication system. *Security and Privacy*, 2(5):e88, 2019.

[ALUK19] Abbas Acar, Long Lu, A Selcuk Uluagac, and Engin Kirda. An analysis of malware trends in enterprise networks. In *International Conference on Information Security*, pages 360–380. Springer, Cham, 2019.

[and] Coinhive blocker project. https://github.com/andreas0607/CoinHive-blocker. Accessed: 2021-2-23.

[ANDB20] Azizah Binti Abdul Aziz, Syahrulanuar Bin Ngah, Yau Ti Dun, and Tan Fui Bee. Coinhive's monero drive-by crypto-jacking. In *IOP Conference Series: Materials Science and Engineering*, volume 769, page 012065. IOP Publishing, 2020.

[ant] Official page of antminingpool web mining pool. https://v3.antpool.com/home. Accessed: 2020-04-13.

[ARI] Jemimah Molina Augusto Remillano II. Coinminer, ddos bot attack docker daemon ports. https://www.trendmicro.com/vinfo/hk-en/security/news/virtualization-and-cloud/coinminer-ddos-bot-attack-docker-daemon-ports. Accessed: 2021-2-14.

[ASKS19] Azuan Ahmad, Wan Shafiuddin, Mohd Nazri Kama, and Madihah Mohd Saudi. A new cryptojacking malware classifier model based on dendritic cell algorithm. In *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, pages 1–5, 2019.

[Ava] Avast. Avastantimalware. https://www.avast.com/c-protect-yourself-from-cryptojacking. Accessed: 2020-04-09.

[Avi] Nadav Avital. New research: Crypto-mining drives almost 90% of all remote code execution attacks. https://www.imperva.com/blog/new-research-crypto-mining-drives-almost-90-remote-code-execution-attacks/. Accessed: 2021-2-23.

[B$^+$14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[BBD19a] Hugo LJ 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)*, pages 1627–1644, 2019.

[BBD19b] Hugo LJ Bijmans, Tim M Booij, and Christian Doerr. Just the tip of the iceberg: Internet-scale exploitation of routers for cryptojacking. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 449–464, 2019.

[BDLF$^+$16] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Natalia Kulatova, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy, et al. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (PLAS)*, pages 91–96, 2016.

[Ben] Víctor labayen (2020). network traffic for machine learning classification dataset. http://doi.org/10.17632/5pmnkshffm.1.

[BI17] Elisa Bertino and Nayeem Islam. Botnets and internet of things security. *Computer*, 50(2):76–79, 2017.

[bit] Bitcoin cash official community page. `https://www.bitcoincash.org/`. Accessed: 2020-04-28.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography Conference*, pages 60–79. Springer, 2006.

[BMZ20] Weikang Bian, Wei Meng, and Mingxue Zhang. Minethrottle: Defending against wasm in-browser cryptojacking. In *Proceedings of The Web Conference (WWW) 2020*, pages 3112–3118, 2020.

[BNM$^+$14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security (ICFCIDS)*, pages 486–504. Springer, 2014.

[Bod] Max Boddy. Phishing attack performed on xrp network. `https://cointelegraph.com/news/phishing-sites-use-trick-letters-in-domain-names-to-steal-xrp`. Accessed: 2020-04-13.

[Bro] The official webpage of browsermine. `https://browsermine.com/`. Accessed: 2021-4-7.

[CAA$^+$19] Z. Berkay Celik, Abbas Acar, Hidayet Aksu, Ryan Sheatsley, Patrick McDaniel, and A. Selcuk Uluagac. Curie: Policy-based secure data exchange. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pages 121–132, 2019.

[CBOS19] Domhnall Carlin, Jonah Burgess, Philip O'Kane, and Sakir Sezer. You could be mine (d): the rise of cryptojacking. *IEEE Security & Privacy*, 18(2):16–22, 2019.

[Cea19] Z. Berkay Celik and et al. Curie: Policy-based secure data exchange. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, page 121–132, 2019.

[CGLP19] Mauro Conti, Ankit Gangwal, Gianluca Lain, and Samuele Giuliano Piazzetta. Detecting covert cryptomining using hpc. *arXiv preprint arXiv:1909.00268*, 2019.

[CGR18] Mauro Conti, Ankit Gangwal, and Sushmita Ruj. On the economic significance of ransomware campaigns: A bitcoin transactions perspective. *Computers & Security*, 79:162–189, 2018.

[Chr] Nilesh Christopher. Hackers mined a fortune from indian websites. https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/hackers-mined-a-fortune-from-indian-websites/articleshow/65836088.cms. Accessed: 2021-2-23.

[Cima] Catalin Cimpanu. Chrome extension caught stealing crypto-wallet keys. https://www.zdnet.com/article/chrome-extension-caught-stealing-crypto-wallet-private-keys/. Accessed: 2020-04-13.

[Cimb] Catalin Cimpanu. A crypto-mining botnet has been hijacking mssql servers for almost two years. https://www.zdnet.com/article/a-crypto-mining-botnet-has-been-hijacking-mssql-servers-for-almost-two-years/. Accessed: 2020-02-17.

[Cimc] Catalin Cimpanu. Mikrotik router hack affect 200k routers in the world. https://www.bleepingcomputer.com/news/security/massive-coinhive-cryptojacking-campaign-touches-over-200-000-mikrotik-routers/. Accessed: 2021-2-23.

[Cla] Thomas Claburn. Google tag manager exploited. https://www.theregister.com/2017/11/22/cryptojackers_google_tag_manager_coin_hive/. Accessed: 2021-02-23.

[CLMS18] Mauro Conti, Qian Qian Li, Alberto Maragno, and Riccardo Spolaor. The dark side(-channel) of mobile devices: A survey on network traffic analysis. *IEEE Communications Surveys Tutorials*, 20(4):2658–2713, 2018.

[clo] Detect large-scale cryptocurrency mining attack against kubernetes clusters. https://azure.microsoft.com/en-us/blog/detect-largescale-cryptocurrency-mining-attack-against-kubernetes-clusters/. Accessed: 2021-2-20.

[Coia] Coinblockerlists. https://zerodot1.gitlab.io/CoinBlockerListsWeb/. Accessed: 2020-06-17.

[Coib] Coinhive blockerproject github page. https://github.com/Marfjeh/coinhive-block. Accessed: 2021-2-23.

[Coic] Coinhive snapshots that taken from webarchive. https://web.archive.org/web/20181101000000*/coinhive.com. Accessed: 2020-05-23.

[Coid] The official webpage of both coinhive and authedmine. http://web.archive.org/web/20190130232758/https://coinhive.com/documentation. Accessed: 2020-10-19.

[Coie] The official webpage of coinhave. http://web.archive.org/web/20180102115842/https://coin-have.com/. Accessed: 2021-4-7.

[Coif] The official webpage of coinimp. https://www.coinimp.com/documentation. Accessed: 2020-10-19.

[Coig] The official webpage of coinnebula. https://web.archive.org/web/20180818144049/https://coinnebula.com/. Accessed: 2021-4-7.

[COSB18] Domhnall Carlin, Philip O'kane, Sakir Sezer, and Jonah Burgess. Detecting cryptomining using dynamic analysis. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018.

[CPR] Check-Point-Research. Checkpoint 2020 cyber security report. https://research.checkpoint.com/2020/the-2020-cyber-security-report/. Accessed: 2021-2-23.

[CRODP19] Maurantonio Caprolu, Simone Raponi, Gabriele Oligeri, and Roberto Di Pietro. Crypto mining makes noise. *arXiv:1910.09272*, 2019.

[crya] Cryptoloot. https://crypto-loot.org/. Accessed: 2020-06-20.

[Cryb] The official webpage of cryptoloot. https://crypto-loot.org/. Accessed: 2021-4-7.

[CS] Cado-Security. Aws cloud-based cryptojacking report. https://www.cadosecurity.com/post/team-tnt-the-first-crypto-mining-worm-to-steal-aws-credentials. Accessed: 2020-02-16.

[Dav] Corbin Davenport. Opera mini and mobile now block cryptocurrency-mining scripts. https://www.androidpolice.com/2018/01/22/opera-mini-mobile-now-block-cryptocurrency-mining-scripts/. Accessed: 2020-10-16.

[Dee] The official github page of deep miner. https://github.com/deepwn/deepMiner. Accessed: 2021-4-7.

[Dep] Statista Research Department. Estimated iot device count by 2025. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/. Accessed: 2021-2-23.

[DHD+20] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza M Parizi, and Kim-Kwang Raymond Choo. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, pages 1–11, 2020.

[DOD] Miner found at usa department of defense. https://www.zdnet.com/article/bug-hunter-finds-cryptocurrency-mining-botnet-on-dod-network/. Accessed: 2020-04-13.

[DSG14] Evan Duffield, Holger Schinzel, and Fernando Gutierrez. Transaction locking and masternode consensus: A mechanism for mitigating double spending attacks. *CryptoPapers. info*, 2014.

[DWA+19] Sanjeev Das, Jan Werner, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 20–38. IEEE, 2019.

[DZG+20] Stanislav Dashevskyi, Yury Zhauniarovich, Olga Gadyatskaya, Aleksandr Pilgun, and Hamza Ouhssain. Dissecting android cryptocurrency miners. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 191–202, 2020.

[Ele] The official webpage of electroneum coin. https://electroneum.com/. Accessed: 2021-4-7.

[ELMC18] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. In *2018 IEEE Eu-*

ropean Symposium on Security and Privacy Workshops (EuroS&PW), pages 58–66. IEEE, 2018.

[Ett] Ettercap project official page. https://www.ettercap-project.org/. Accessed: 2021-06-04.

[GJPS18] A Goriacheva, N Jakubenko, O Pogodina, and D Silnov. Anonymization technologies of cryptocurrency transactions as money laundering instrument. *KnE Social Sciences*, pages 46–53, 2018.

[goo] Google bans crypto-mining apps from play store. https://www.bbc.com/news/technology-44980936. Accessed: 2020-10-16.

[Gre] Larry Greenemeier. Crytojacking can corrupt the iot. https://www.scientificamerican.com/article/how-cryptojacking-can-corrupt-the-internet-of-things/. Accessed: 2021-2-23.

[Gru] Josh Grunzweig. Large scale monero mining operation. https://unit42.paloaltonetworks.com/unit42-large-scale-monero-cryptocurrency-mining-operation-using-xmrig/. Accessed: 2021-2-23.

[GTBS12] Claudio Guarnieri, Alessandro Tanasi, Jurriaan Bremer, and Mark Schloesser. The cuckoo sandbox. 2012. https://www.cuckoosandbox.org.

[Hac] Robert Hackett. Tesla hackers hacked aws cloud services to mine monero. https://fortune.com/2018/02/20/tesla-hack-amazon-cloud-cryptocurrency-mining/. Accessed: 2020-10-19.

[HAL+18] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631. IEEE, 2018.

[HDM+14] Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C Snoeren, and Kirill Levchenko. Botcoin: Monetizing stolen cycles. In *Network and Distributed Systems Security (NDSS) Symposium*. Citeseer, 2014.

[HL15] Ryan Heartfield and George Loukas. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys (CSUR)*, 48(3):1–39, 2015.

[HPV+20] Ralph Holz, Diego Perino, Matteo Varvello, Johanna Amann, Andrea Continella, Nate Evans, Ilias Leontiadis, Christopher Natoli, and Quirin Scheitle. A retrospective analysis of user exposure to (illicit) cryptocurrency mining on the web. *arXiv preprint arXiv:2004.13239*, 2020.

[HRS+17] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with webassembly. page 185–200, 2017.

[HYY+18] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Haixin 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)*, pages 1701–1713, 2018.

[iMSVBR19] Jordi Zayuelas i Muñoz, José Suárez-Varela, and Pere Barlet-Ros. Detecting cryptocurrency miners with netflow/ipfix network measurements. In *2019 IEEE International Symposium on Measurements & Networking*, pages 1–6, 2019.

[IoT] Security intelligence, liquor iot botnet. https://securityintelligence.com/news/weekly-security-news-roundup-mirai-inspired-liquorbot-botnet-mining-for-monero/. Accessed: 2021-07-04.

[IS] IBM-Security. X-force threat intelligence index 2020. https://securityintelligence.com/series/ibm-x-force-threat-intelligence-index-2020. Accessed: 2021-02-16.

[JP20] Keshani Jayasinghe and Guhanathan Poravi. A survey of attack instances of cryptojacking targeting cloud infrastructure. In *Proceedings of the 2020 2nd Asia Pacific Information Technology Conference*, pages 100–107, 2020.

[JSE] Jse coin official webpage. https://jsecoin.com/. Accessed: 2021-4-7.

[Kasa] Kaspersky. Kaspersky global reports. https://www.kaspersky.com/about/press-releases/2019_fear-of-the-unknown. Accessed: 2020-10-19.

[Kasb] Limor Kassem. xmrig father zeus of cryptocurrency mining malware. https://securityintelligence.com/xmrig-father-zeus-of-cryptocurrency-mining-malware/. Accessed: 2021-2-23.

[KBRS20] Conor Kelton, Aruna Balasubramanian, Ramya Raghavendra, and Mudhakar Srivatsa. Browser-based deep behavioral detection of web cryptomining with coinspy. In *27th Annual Network and Distributed System Security Symposium, NDSS*, pages 23–26, 2020.

[Kea20] Ahmet Kurt and et al. Lnbot: A covert hybrid botnet on bitcoin lightning network for fun and profit. In *Computer Security – ESORICS 2020*, pages 734–755, Cham, 2020. Springer International Publishing.

[Ken] Emma Kent. Miner found in popular game in steam. https://www.eurogamer.net/articles/2018-07-30-steam-game-abstractism-turns-pcs-into-cryptocurrency-miners. Accessed: 2020-04-19.

[KMM$^+$19] 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)*, pages 840–852, 2019.

[KRB$^+$15] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pages 3–24. Springer, 2015.

[KVM$^+$18] 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)*, pages 1714–1730, 2018.

[Lab68] Sanford Labovitz. Criteria for selecting a significance level: A note on the sacredness of. 05. *The American Sociologist*, pages 220–222, 1968.

[LEBM20] Nada Lachtar, Abdulrahman Abu Elkhail, Anys Bacha, and Hafiz Malik. A cross-stack approach towards defending against cryptojacking. *IEEE Computer Architecture Letters*, 19(2):126–129, 2020.

[Lit] Litecoin official webpage. `https://litecoin.org/`. Accessed: 2021-2-23.

[Loo] Lookout. Fake bitcoin wallet. `https://blog.lookout.com/fake-bitcoin-wallet`. Accessed: 2020-04-13.

[LYK+19] Seunghyeon Lee, Changhoon Yoon, Heedo Kang, Yeonkeun Kim, Yongdae Kim, Dongsu Han, Sooel Son, and Seungwon Shin. Cybercriminal minds: An investigative study of cryptocurrency abuses in the dark web. In *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[LZC+18] Jingqiang Liu, Zihao Zhao, Xiang Cui, Zhi Wang, and Qixu Liu. A novel approach for detecting browser-based silent miner. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 490–497. IEEE, 2018.

[MA] Dave McMillen and Michelle Alvarez. Mirai iot botnet: Mining for bitcoins? `https://securityintelligence.com/mirai-iot-botnet-mining-for-bitcoins/`. Accessed: 2021-2-23.

[Mar] Andrew Marshall. Combined crypto market capitalization races past $800 bln. `https://cointelegraph.com/news/combined-crypto-market-capitalization-races-past-800-bln`. Accessed: 2020-02-28.

[McD] Craig McDonald. Cryptojacking malware hid into emails. `https://www.mailguard.com.au/blog/brandjacking-malware-hiding`. Accessed: 2021-02-23.

[McM] Dave McMillen. What is the mirai botnet? `https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/`. Accessed: 2020-11-02.

[Mil] Annaliese Milano. Russian scientists arrested crypto mining nuclear lab. https://www.coindesk.com/russian-scientists-arrested-crypto-mining-nuclear-lab. Accessed: 2021-2-23.

[Mina] Minerblock: An efficient browser extension to block browser-based cryptocurrency miners all over the web. https://github.com/xd4rker/MinerBlock/blob/master/assets/filters.txt. Accessed: 2020-04-08.

[minb] Official page of minergate web mining pool. https://www.minergate.com. Accessed: 2020-04-13.

[Min18] Miners in youtube ads. https://arstechnica.com/information-technology/2018/01/now-even-youtube-serves-ads-with-cpu-draining-cryptocurrency-miners/, Jan 2018. Accessed: 2020-04-13.

[MJS19] Per Håkon Meland, Bent Heier Johansen, and Guttorm Sindre. An experimental analysis of cryptojacking attacks. In *Nordic Conference on Secure IT Systems*, pages 155–170. Springer, 2019.

[Mona] Monero price chart. https://coinmarketcap.com/currencies/monero/. Accessed: 2020-04-13.

[Monb] Official page of monero cryptocurrency and its white paper. https://www.getmonero.org/resources/research-lab/. Accessed: 2020-02-16.

[Monc] The official webpage of monerise. http://web.archive.org/web/20200813110918/http://monerise.com/. Accessed: 2021-4-7.

[MPB$^+$20] Ganapathy Mani, Vikram Pasumarti, Bharat Bhargava, Faisal Tariq Vora, James MacDonald, Justin King, and Jason Kobes. Decrypto pro: Deep learning based cryptomining malware detection using performance counters. In *IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 109–118. IEEE, 2020.

[MSH$^+$18] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2018(3):143–163, 2018.

[MV] Byron Gelera Mark Vicente, Johnlery Triunfante. Cve-2019-2725 exploited, used to deliver monero miner. https://www.trendmicro.com/en_ca/research/19/f/cve-2019-2725-exploited-and-certificate-files-used-for-obfuscation-to-deliver-monero-miner.html. Accessed: 2021-2-23.

[MWJR18] Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck. Web-based cryptojacking in the wild. *arXiv preprint arXiv:1808.09474*, 2018.

[MWJR19] Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck. Thieves in the browser: Web-based cryptojacking in the wild. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES)*, pages 1–10, 2019.

[NAB+21] Faraz Naseem, Ahmet Aris, Leonardo Babun, Ege Tekiner, and Selcuk Uluagac. MINOS: A lightweight real-time cryptojacking detection system. In *28th Annual Network and Distributed System Security Symposium, NDSS, February 21-25, 2021*, 2021.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system., 2008.

[Ner] The official webpage of nerohut. https://web.archive.org/web/20190131001253/https://nerohut.com/documentation.php. Accessed: 2020-10-19.

[New] Lily Hay Newman. Google bans all cryptomining extensions from the chrome store. https://www.wired.com/story/google-bans-all-cryptomining-extensions-from-the-chrome-store/. Accessed: 2020-10-16.

[NKK10] Seung Yeob Nam, Dongwon Kim, and Jeongeun Kim. Enhanced arp: preventing arp poisoning-based man-in-the-middle attacks. *IEEE communications letters*, 14(2):187–189, 2010.

[NLFM20] Helio N Cunha Neto, Martin Andreoni Lopez, Natalia C Fernandes, and Diogo MF Mattos. Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking. *Annals of Telecommunications*, pages 1–11, 2020.

[NoC] Nocoin: Block lists to prevent javascript miners. https://github.com/hoshsadiq/adblock-nocoin-list. Accessed: 2020-04-08.

[Nor] Norton. Official site — norton™ - antivirus, anti-malware software. https://us.norton.com/. Accessed: 2020-04-09.

[NWX$^+$19] Rui Ning, Cong Wang, ChunSheng Xin, Jiang Li, Liuwan Zhu, and Hongyi Wu. Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1873–1881. IEEE, 2019.

[OALU21] Harun Oz, Ahmet Aris, Albert Levi, and A Selcuk Uluagac. A survey on ransomware: Evolution, taxonomy, and defense solutions. *arXiv preprint arXiv:2102.06249*, 2021.

[Ole] Doug Olenick. Miner into third party zoom. https://www.trendmicro.com/en_us/research/20/d/zoomed-in-a-look-into-a-coinminer-bundled-with-zoom-installer.html. Accessed: 2020-04-13.

[Osb18] Charlie Osborne. Apple bans developers from submitting cryptocurrency mining apps for ios devices. https://www.zdnet.com/article/apple-bans-developers-from-creating-ios-cryptocurrency-mining-apps/, 2018. Accessed: 2020-10-16.

[Para] Darren Parkin. Cryptocurrency stealer malware. https://www.express.co.uk/finance/city/1213514/cryptocurrency-fraud-malware-clipper-victims-crv. Accessed: 2020-04-13.

[Parb] Helen Partz. Ukrainian man faces up to 6 years in jail for cryptojacking on his own websites. https://cointelegraph.com/news/ukrainian-man-faces-up-to-6-years-in-jail-for-cryptojacking-on-his-own-websites. Accessed: 2021-2-23.

[PCHD19] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[Pea] Jordan Pearson. A 'fortnite' cheat maker duped players into downloading a bitcoin miner. https://www.vice.com/en/article/8x598p/a-fortnite-cheat-maker-duped-players-into-downloading-a-bitcoin-miner-epic-games-sued. Accessed: 2021-2-23.

[Phi] Phishing attack caused 1.7 billion loss. https://cointelegraph.com/news/israeli-citizen-accused-of-stealing-over-17-million-in-crypto. Accessed: 2020-04-13.

[PIB20] Ivan Petrov, Luca Invernizzi, and Elie Bursztein. Coinpolice: Detecting hidden cryptojacking attacks with neural networks. *arXiv:2006.10861*, 2020.

[PIM19] Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos Markatos. Truth in web mining: Measuring the profitability and the imposed overheads of cryptojacking. In *International Conference on Information Security (ISC)*, pages 277–296. Springer, 2019.

[Por] General explanation of port mirroring. https://www.miarec.com/faq/what-is-port-mirroring. Accessed: 2021-06-04.

[PST19] Sergio Pastrana and Guillermo Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 73–86, 2019.

[pub] Source code search engine. https://publicwww.com/. Accessed: 2020-10-16.

[Ran] Implementation of randomx proof of work algorithm. https://github.com/tevador/RandomX.

[Red] Redactie. Analysing a cryptocurrency phishing attack that earns $15k in two hours. https://www.kpn.com/zakelijk/blog/analysing-cryptocurrency-phishing-attack.htm. Accessed: 2020-04-13.

[Res] Check Point Research. Cloud-based cryptojacking article. https://research.checkpoint.com/2020/the-2020-cyber-security-report/. Accessed: 2020-10-19.

[Ret] Retro cryptomining project github page. https://github.com/retrocryptomining/data. Accessed: 2021-2-23.

[RMY20] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. Blag: Improving the accuracy of blacklists. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020.

[RP18] Juan D Parra Rodriguez and Joachim Posegga. Rapid: Resource and api-based detection against in-browser miners. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 313–326, 2018.

[RS] Kimberly Goody Rakesh Sharma, Akhil Reddy. A vulnerability used to deliver cryptojacking malware. https://www.fireeye.com/blog/threat-research/2018/02/cve-2017-10271-used-to-deliver-cryptominers.html. Accessed: 2021-02-23.

[RS19] Muhammad Amirrudin Razali and Shafiza Mohd Shariff. Cmblock: In-browser detection and prevention cryptojacking tool using blacklist and behavior-based detection method. In *International Visual Informatics Conference (IVIC)*, pages 404–414. Springer, 2019.

[RSD+18] 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)*, pages 1–10, 2018.

[RZWH18] Jan Rüth, Torsten Zimmermann, Konrad Wolsing, and Oliver Hohlfeld. Digging into browser-based crypto mining. In *Proceedings of the Internet Measurement Conference (IMC) 2018*, pages 70–76, 2018.

[Sal19] Salon. Faq: What happens when i choose to “suppress ads” on salon? https://web.archive.org/web/20200604052723if_/https://www.salon.com/about/faq-what-happens-when-i-choose-to-suppress-ads-on-salon, 2019. Accessed: 2020-10-16.

[San] Maria Santos. utorrent update smuggles shady cryptocurrency miner into your computer. https://99bitcoins.com/utorrent-update-cryptocurrency-miner/. Accessed: 2020-03-31.

[SC] Silvia Sebastián and Juan Caballero. Brilliant but evil: Gaming company fined $1 million for secretly using players’ computers to mine bitcoin. https://www.forbes.com/sites/

kashmirhill/2013/11/19/brilliant-but-evil-gaming-company-turned-players-computers-into-unwitting-bitcoin-mining-slaves/?sh=11f7e958570b. Accessed: 2021-2-23.

[Sch] Mathew J. Schwartz. Social engineering attacks for cryptojacking. https://www.bankinfosecurity.com/cryptocurrency-theft-hackers-repurpose-old-tricks-a-10685. Accessed: 2021-2-23.

[Sch96] Mark J Schervish. P values: what they are and what they are not. *The American Statistician*, 50(3):203–206, 1996.

[sci] Official page of scikit-learn python library. https://scikit-learn.org/stable/. Accessed: 2021-07-22.

[SE11] Aditya K Sood and Richard J Enbody. Malvertising–exploiting web advertising. *Computer Fraud & Security*, 2011(4):11–16, 2011.

[Sea] Tara Seals. Aws cloud-based cryptojacking report. https://threatpost.com/aws-cryptojacking-worm-cloud/158427/. Accessed: 2021-2-23.

[Sig18] Karl Sigler. Crypto-jacking: how cyber-criminals are exploiting the crypto-currency boom. *Computer Fraud & Security*, 2018(9):12–14, 2018.

[SKM19] Muhammad Saad, Aminollah Khormali, and Aziz Mohaisen. Dine and dash: Static, dynamic, and economic analysis of in-browser cryptojacking. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12. IEEE, 2019.

[Smi] Trevor Smith. Miner found at nintendo switch console. https://bitcoinist.com/nintendo-switch-game-pulled-over-cryptojacking-concerns/. Accessed: 2020-04-13.

[son20] Mid-year update — july 2020, 2020 sonicwall cyber threat report. https://www.sonicwall.com/resources/2020-cyber-threat-report-mid-year-update-pdf/, July 2020. Accessed: 2020-10-19.

[Stea] Lukas Stefanko. Fake cryptocurrency apps google play bitcoin. https://www.welivesecurity.com/2019/05/23/fake-cryptocurrency-apps-google-play-bitcoin/. Accessed: 2020-04-13.

[Steb] Joe Stewart. Cryptocurrency-stealing malware landscape. `https://www.secureworks.com/research/cryptocurrency-stealing-malware-landscape`. Accessed: 2020-04-13.

[Sum] The official webpage of sumokoin. `https://www.sumokoin.org/`. Accessed: 2021-4-7.

[Tan20] Dmitry Tanana. Behavior-based detection of cryptojacking malware. In *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 0543–0545. IEEE, 2020.

[TAU$^+$21] Ege Tekiner, Abbas Acar, A. Selcuk Uluagac, Engin Kirda, and Ali Aydin Selcuk. Sok: Cryptojacking malware. In *6th IEEE European Symposium on Security and Privacy*, Virtual, September 2021.

[TDA$^+$19] Rashid Tahir, Sultan Durrani, Faizan Ahmed, Hammas Saeed, Fareed Zaffar, and Saqib Ilyas. The browsers strike back: countering cryptojacking and parasitic miners on the web. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 703–711. IEEE, 2019.

[TNL18] Hironao Takahashi, Shinji Nakano, and Uzair Lakhani. Sha256d hash rate enhancement by l3 cache. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 849–850. IEEE, 2018.

[Tre] Trend micro iot botnet article. `https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/into-the-battlefield-a-security-guide-to-iot-botnets`. Accessed: 2021-07-04.

[tsfa] Tsfresh official features page. `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`.

[tsfb] Tsfresh official page. `https://tsfresh.readthedocs.io/en/latest/`.

[tsr] Tefresh relevance table official documents. `https://tsfresh.readthedocs.io/en/latest/_modules/tsfresh/feature_selection/relevance.html`. Accessed: 2021-02-23.

[UKG18] Uk government plugin based mining. `https://www.digitaltrends.com/computing/government-websites-plugin-coinhive-monero-miner/`, Feb 2018. Accessed: 2020-04-13.

[UNIa] UNICEF. Give hope, just by being here. https://web.archive.org/web/20200518224428/https://www.thehopepage.org/. Accessed: 2020-10-16.

[Unib] Unit42. Watchdog: Exposing a cryptojacking campaign that's operated for two years. https://unit42.paloaltonetworks.com/watchdog-cryptojacking/. Accessed: 2021-02-23.

[Upl] The official webpage of uplexa coin. https://uplexa.com/. Accessed: 2021-4-7.

[Vas14] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. *https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf*, 71, 2014.

[VGOB20] Said Varlioglu, Bilal Gonen, Murat Ozer, and Mehmet Bastug. Is cryptojacking dead after coinhive shutdown? In *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, pages 385–389. IEEE, 2020.

[VS13] Nicolas Van Saberhagen. Cryptonote v 2.0. https://bytecoin.org/old/whitepaper.pdf, 2013. Accessed: 2021-02-23.

[was] Webassembly. https://webassembly.org//. Accessed: 2021-02-23.

[weba] Domain data & threat intel, powered by artificial intelligence. https://www.webshrinker.com/. Accessed: 2020-10-16.

[webb] Internet archive wayback machine. https://web.archive.org/. Accessed: 2020-10-16.

[Webc] Lg webos development framework. https://webostv.developer.lge.com/. Accessed: 2020-02-16.

[webd] The official webpage of webmine. http://webmine.cz/. Accessed: 2021-4-7.

[webe] The official webpage of webmine pool. https://www.webminepool.com/. Accessed: 2021-4-7.

[webf] The official webpage of webminepool. https://www.webminepool.com/. Accessed: 2021-4-7.

[Webg] The official webpage of webminerpool. https://github.com/notgiven688/webminerpool. Accessed: 2020-10-19.

[WFX+18] Wenhao Wang, Benjamin Ferrell, Xiaoyang Xu, Kevin W Hamlen, and Shuang Hao. Seismic: Secure in-lined script monitors for interrupting cryptojacks. In *European Symposium on Research in Computer Security*, pages 122–142, 2018.

[Win] Andrew Windsor. Breaking down a two-year run of vivin's cryptominers. https://blog.talosintelligence.com/2020/01/vivin-cryptomining-campaigns.html. Accessed: 2021-2-20.

[wir] Wireshark. https://www.wireshark.org/. Accessed: 2021-07-21.

[Won] Joon Ian Wong. An italian bank's server was hijacked to mine bitcoin. https://qz.com/1024930/bitcoin-malware-an-italian-banks-server-was-hijacked-to-mine-bitcoin-says-darktrace/. Accessed: 2020-02-17.

[Wor] Wordpress official page. https://wordpress.com/. Accessed: 2021-06-04.

[XMR] Xmrrig. https://github.com/xmrig/xmrig. Accessed: 2021-2-23.

[YRRR12] Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/Acm Transactions on Networking*, 20(5):1663–1677, 2012.

[YSWAM19] Arief Dwi Yulianto, Parman Sukarno, Aulia Arif Warrdana, and Muhammad Al Makky. Mitigation of cryptojacking attacks using taint analysis. In *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 234–238. IEEE, 2019.

[ZWM19] Aaron Zimba, Zhaoshun Wang, and Mwenge Mulenga. Cryptojacking injection: A paradigm shift to cryptocurrency-based web-centric internet attacks. *Journal of Organizational Computing and Electronic Commerce*, 29(1):40–59, 2019.

[ZWMO20] Aaron Zimba, Zhaoshun Wang, Mwenge Mulenga, and Nickson Herbert Odongo. Crypto mining attacks in information systems: An emerging threat to cyber security. *Journal of Computer Information Systems*, 60(4):297–308, 2020.

VITA

EGE TEKINER

2019	B.S., Computer Engineering Ankara University Ankara, Turkey
2017 - 2021	Solution Architech Paribu Cryptocurrency Exchange Platform Istanbul, Turkey
2020 - 2021	M.S., Electrical and Computer Engineering Florida International University Miami, Florida

SELECTED PUBLICATIONS

Ege Tekiner, Abbas Acar, Engin Kirda, Ali Aydin Selcuk, A. Selcuk Uluagac, *"SoK: Cryptojacking Malware"*, in 6th IEEE European Symposium on Security and Privacy, EuroS&P, September 6-10, 2021.

Ege Tekiner, Abbas Acar, Engin Kirda, Ali Aydin Selcuk, A. Selcuk Uluagac, *"In-Browser Cryptomining for Good: An UntoldStory"*, The 3rd IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) August 23-26, 2021.

Ege Tekiner, Abbas Acar, A. Selcuk Uluagac, *A Lightweight IoT Cryptojacking DetectionMechanism in Heterogeneous Smart HomeNetworks"*, in 29th Annual Network and Distributed System Security Symposium, NDSS, 27 February – 3 March 2022. (Under Review)

F. Naseem, A. Aris, L. Babun, **Ege Tekiner**, and S. Uluagac, "MINOS: A lightweight real-time cryptojacking detection system," in 28th Annual Network and Distributed System Security Symposium, NDSS, February 21-25, 2021, 2021.