MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense

Radhesh Krishnan Konoth
Vrije Universiteit Amsterdam
r.k.konoth@vu.nl

Emanuele Vineti
Vrije Universiteit Amsterdam
emanuele.vineti@gmail.com

Veelasha Moonsamy
Utrecht University
email@veelasha.org

Martina Lindorfer
TU Wien
martina@iseclab.org

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

Herbert Bos
Vrije Universiteit Amsterdam
herbertb@cs.vu.nl

Giovanni Vigna
UC Santa Barbara
vigna@cs.ucsb.edu

ABSTRACT

A wave of alternative coins that can be effectively mined without specialized hardware, and a surge in cryptocurrencies' market value has led to the development of cryptocurrency mining (*cryptomining*) services, such as Coinhive, which can be easily integrated into websites to monetize the computational power of their visitors. While legitimate website operators are exploring these services as an alternative to advertisements, they have also drawn the attention of cybercriminals: *drive-by mining* (also known as *cryptojacking*) is a new web-based attack, in which an infected website secretly executes JavaScript code and/or a WebAssembly module in the user's browser to mine cryptocurrencies without her consent.

In this paper, we perform a comprehensive analysis on Alexa's Top 1 Million websites to shed light on the prevalence and profitability of this attack. We study the websites affected by drive-by mining to understand the techniques being used to evade detection, and the latest web technologies being exploited to efficiently mine cryptocurrency. As a result of our study, which covers 28 Coinhive-like services that are widely being used by drive-by mining websites, we identified 20 active cryptomining campaigns.

Motivated by our findings, we investigate possible countermeasures against this type of attack. We discuss how current blacklisting approaches and heuristics based on CPU usage are insufficient, and present MineSweeper, a novel detection technique that is based on the intrinsic characteristics of cryptomining code, and, thus, is resilient to obfuscation. Our approach could be integrated into browsers to warn users about silent cryptomining when visiting websites that do not ask for their consent.

CCS CONCEPTS

• **Security and privacy → Browser security**; **Malware and its mitigation**; • **Social and professional topics →** *Computer crime*;

KEYWORDS

cryptocurrency; mining; cryptojacking; drive-by attacks; malware

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *CCS '18: 2018 ACM SIGSAC Conference on Computer & Communications Security Oct. 15–19, 2018, Toronto, ON, Canada*, https://doi.org/10.1145/3243734.3243858.

ACM Reference Format:
Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. 2018. MineSweeper: An In-depth Look into Drive-by, Cryptocurrency Mining and Its Defense. In *CCS '18: 2018 ACM SIGSAC Conference on Computer & Communications Security Oct. 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3243734.3243858

1 INTRODUCTION

Ever since its introduction in 2009, Bitcoin [47] has attracted the attention of cybercriminals due to the possibility to perform and receive anonymous payments. In addition, the financial reward for using computing power for mining has incentivized criminals to experiment with *silent* cryptocurrency miners (*cryptominers*), which gained popularity among malware authors who were, after all, already in the business of compromising PCs and herding large numbers of them in botnets. However, as Bitcoin mining became too difficult for regular machines, the profits of mining botnets dwindled, and Bitcoin-mining botnets declined: an analysis by McAfee in 2014 suggested that malicious miners are not profitable on PCs and certainly not on mobile devices [37].

Since then, a wave of alternative coins (altcoins) has been introduced: the market now counts over 1,500 cryptocurrencies, out of which more than 600 see an active trade. At the time of writing, they represent over 50% of the cryptocurrency market [24]. Unlike Bitcoin, many of them are still mineable without specialized hardware. Furthermore, miners can organize themselves into *mining pools*, which allow members to distribute mining tasks and share the rewards. These new currencies, and an overall surge in market value across cryptocurrencies at the end of 2017 [26], has renewed interest in cryptominers and led to the proliferation of cryptomining services, such as Coinhive [5], which can easily be integrated into a website to mine on its visitors' devices from within the browser.

For cybercriminals, these services provide a low-effort way to monetize websites as part of *drive-by mining* (or *cryptojacking*) attacks: they either compromise webservers (through exploits [15, 39, 50, 62, 65], or taking advantage of misconfigurations [49]) and install JavaScript-based miners, distribute their miners through advertisements (including Google's DoubleClick on YouTube [28] and the AOL advertising platform [41]), or compromise third-party libraries [71] included in numerous websites. Attackers also have come up with creative tactics to conceal their attack, for example by using "pop-under" windows [27] (to maximize the time a victim spends on the mining website), or by abusing Coinhive's URL shortening service [77]. Finally, rogue WiFi hotspots [20] and compromised routers [35] allow attackers to inject the mining payload on a large scale into *any* website that their users visit.

However, in-browser mining is not malicious per-se: charities, such as UNICEF [40], launched dedicated websites to mine for donations, and legitimate websites are exploring mining in an attempt to monetize their content in the presence of ad blockers [58]. Whether users accept cryptocurrency miners as an alternative to invasive advertisements, which raise privacy concerns due to wide-spread targeting and tracking [19, 43, 52], remains to be seen. For them, in-browser mining degrades their system's performance and increases its power consumption [51]. Therefore, the key distinction between these use cases and drive-by mining attacks is user consent and whether a website discloses its mining activity or not. For example, as a way to enforce user consent for in-browser mining, Coinhive launched AuthedMine [6], which explicitly requires user input. However, a related study has found that this API has not yet found widespread adoption [60]. Related work also suggested the introduction of a "do not mine" HTTP header [25], which, however, websites do not necessarily need to honor.

To study the prevalence of drive-by mining attacks, i.e., in-browser mining without requiring any user interaction or consent, we performed a comprehensive analysis of Alexa's Top 1 Million websites [3]. As a result of our study, which covers 28 Coinhive-like services, we identified 20 active cryptomining campaigns. In contrast to a previous study, which found cryptomining on low-value targets, such as parked websites, and concluded that cryptomining was not very profitable [25], we find that cryptomining can indeed make economic sense for an attacker. We identified several video players used by popular video streaming websites that include cryptomining code and which maximize the time users spend on a website mining for the attacker—potentially earning more than US$ 30,000 a month. Furthermore, we found that instead of JavaScript-based attacks, drive-by mining now largely takes advantage of WebAssembly (Wasm) to efficiently mine cryptocurrencies and maximize profits.

As a countermeasure, browsers [21, 67, 73], dedicated browser extensions [10, 11], and ad blockers have started to use blacklists. However, maintaining a complete blacklist is not scalable, and it is prone to false negatives. These blacklists are often manually compiled and are easily defeated by URL randomization [59] and domain generation algorithms (DGAs), which are already actively being used in the wild [74]. Other detection attempts look for high CPU usage as an indicator that cryptocurrency mining is taking place. This not only causes false positives for other CPU-intensive use cases, but also causes false negatives, as cryptocurrency miners have started to throttle their CPU usage to evade detection [25].

In this work, we focus on Wasm-based mining, the most efficient and widespread technique for drive-by mining attacks. We propose MineSweeper, a drive-by mining defense that is based on identifying the intrinsic characteristics of the mining itself: the execution of its hashing function. Our first approach is to perform static analysis on the Wasm code and to identify the hashing code based on the cryptographic operations it performs. Currently, attackers avoid heavy obfuscation of the Wasm code as it comes with performance penalties, and hence decreases profits. To deal with future evasion techniques, we present a second, more obfuscation-resilient detection approach: by monitoring CPU cache events at run time we can identify cryptominers based on their memory access patterns.

As browsers are currently struggling to find a suitable alternative to blacklists [29], the techniques used by MineSweeper could be adopted as a defense mechanism against drive-by mining, for example by warning users and enforcing their consent before allowing mining scripts to execute or blocking mining scripts altogether.

In summary, we make the following contributions:

- We perform the first in-depth assessment of drive-by mining.
- We discuss why current defenses based on blacklisting and CPU usage are ineffective.
- We propose MineSweeper, a novel detection approach based on the identification of cryptographic functions through static analysis and monitoring of cache events during run time.

In the spirit of open science, we make the collected datasets and the code we developed for this work publicly available at `https://github.com/vusec/minesweeper`.

2 BACKGROUND

A cryptocurrency is a medium of exchange much like the Euro or the US Dollar, except that it uses cryptography and blockchain technology to control the creation of monetary units and to verify the transaction of a fund. *Bitcoin* [47] was the first such decentralized digital currency. A cryptocurrency user can transfer money to another user by forming a transaction record and committing it to a distributed write-only database called *blockchain*. The blockchain is maintained by a peer-to-peer network of *miners*. A miner collects transaction data from the network, validates it, and inserts it into the blockchain in the form of a block. When a miner successfully adds a valid block to the blockchain, the network compensates the miner with cryptocurrency (e.g., Bitcoins). In the case of Bitcoin, this process is called *Bitcoin mining*, and this is how new Bitcoins enter circulation. Bitcoin transactions are protected with cryptographic techniques that ensure only the rightful owner of a Bitcoin wallet address can transfer funds from it.

To add a block (i.e., a collection of transaction data) to the blockchain, a miner has to solve a cryptographic puzzle based on the block. This mechanism prevents malicious nodes from trying to add bogus blocks to the blockchain and earn the reward illegitimately. A valid block in the blockchain contains a solution to a cryptographic puzzle that involves the hash of the previous block, the hash of the transactions in the current block, and a wallet address to credit with the reward.

2.1 Cryptocurrency Mining Pools

The cryptographic puzzle is designed such that the probability of finding a solution for a miner is proportional to the miner's computational power. Due to the nature of the mining process, the interval between mining events exhibits high variance from the point of view of a single miner. Consequently, miners typically organize themselves into *mining pools*. All members of a pool work together to mine each block, and share the reward when one of them successfully mines a block.

The protocol used by miners to reliably and efficiently fetch jobs from mining pool servers is known as *Stratum* [63]. It is a cleartext communication protocol built over TCP/IP, using a JSON-RPC format. Stratum prescribes that miners who want to join the mining pool first send a *subscription* message, describing the miner's capability in terms of computational resources. The pool server then responds with a *subscription response* message, and the miner sends an *authorization request* message with its username and password. After successful authorization, the pool sends a *difficulty notification* that is proportional to the capability of the miner—ensuring that low-end machines get easier jobs (i.e., puzzles) than high-end ones. Finally, the pool server assigns these jobs by means of *job notifications*. Once the miner finds a solution, it sends it to the pool server in the form of a *share*. The pool server rewards the miner in proportion to the number of valid shares it submitted and the difficulty of the jobs.

2.2 In-browser Cryptomining

The idea of cryptomining by simply loading a webpage using JavaScript in a browser exists since Bitcoin's early days. However, with the advent of GPU- and ASIC-based mining, browser-based Bitcoin mining, which is $1.5x$ slower than native CPU mining [25], became unprofitable. Recently, the cause for the decline of JavaScript-based cryptocurrency miners has subsided: due to new CPU-mineable altcoins and increasing cryptocurrency market value, it is now profitable to mine cryptocurrencies with regular CPUs again. In 2017, Coinhive was the first to revisit the idea of in-browser mining. They provide APIs to website developers for implementing in-browser mining on their websites and to use their visitors' CPU resources to mine the altcoin Monero. Monero employs the CryptoNight algorithm [61] as its cryptographic puzzle, which is optimized towards mining by regular CPUs and provides strong anonymity; hence, it is ideal for in-browser cryptomining.[1] Moreover, the development of new web technologies that have been happening in parallel allows for more efficient—and thus profitable—mining in the browser.

2.3 Web Technologies

Web developers continuously strive to deploy performance-critical parts of their application in the form of native code and run it inside the browser securely. As such, there are on-going research and development efforts to improve the performance of native code execution in the web browser [32, 68]. Naturally, the developers of JavaScript-based cryptominers started exploiting these advancements in web technologies to speed up drive-by mining, thus taking advantage of two web technologies: *asm.js* and *WebAssembly*.

In 2013, Mozilla introduced asm.js, which takes C/C++ code to generate a subset of JavaScript code with annotations that the JavaScript engine can later compile to native code. To improve the performance of native code in the browser even further, in 2017, the World Wide Web Consortium developed WebAssembly (Wasm). Any C/C++/Rust-based application can be easily converted to Wasm, a binary instruction format for a stack-based virtual machine, and executed in the browser at native speed by taking advantage of standard hardware capabilities available on a wide range of platforms. Today, all four major browsers (Firefox, Chrome, Safari, and Edge) support Wasm.

The main difference between *asm.js* and *Wasm* is in the way in which the code is optimized. In asm.js, the JavaScript Just-in-Time (JIT) compiler of the browser converts the JavaScript to an Abstract Syntax Tree (AST). Then, it compiles the AST to non-optimized native code. Later, at run time, the JavaScript JIT engine looks for slow code paths and tries to re-optimize this code at run time. The detection and re-optimization of slow code paths consume a substantial amount of CPU cycles. In contrast, Wasm performs the optimization of the whole module only once, at compile time. As a result, the JIT engine does not need to parse and analyze the Wasm module to re-optimize it. Rather, it directly compiles the module to native code and starts executing it at native speed.

2.4 Existing Defenses against Drive-by Mining

Until now, there is no reliable mechanism to detect drive-by mining. The developers of CoinBlockerLists [4] maintain a blacklist of mining pools and proxy servers that they manually collect from reports on security blogs and Twitter. Dr. Mine [8] attempts to block drive-by mining by means of explicitly blacklisted URLs (based on for example CoinBlockerLists). In particular, it detects JavaScript code that tries to connect to blacklisted mining pools. MinerBlock [10] further combines blacklists with detecting potential mining code inside loaded JavaScript files. Both approaches suffer from high false negatives: as we show in our analysis, most of the drive-by mining websites are using obfuscated JavaScript and randomized URLs to evade the aforementioned detection techniques.

Google engineers from the Chromium project recently acknowledged that blacklisting does not work and that they are looking for alternatives [29]. Specifically, they considered adding an extra permission to the browser to throttle code that runs the CPU at high load for a certain amount of time. Related studies also found high CPU usage from the website as an indicator of drive-by mining [46]. At the same time, another recent study shows that many drive-by miners are throttling their CPU usage to around 25% [25] and simply considering the CPU usage alone as the indicator of drive-by mining suffers from high false negatives. Even without taking the CPU throttling to such extremes, drive-by miners can blend in with other browsing activity, potentially leading to false positives for other CPU-intensive use cases, such as games [59].

Making matters worse, in-browser mining service providers, such as Coinhive, have no incentives to disrupt drive-by mining attacks: Coinhive keeps 30% of the cryptocurrency that is mined with its code. In reaction to abuse complaints, they reportedly keep all of the profits of campaigns, whose members still keep mining cryptocurrency even after their site key (i.e., the campaign's account identifier with Coinhive) has been terminated [36].

3 THREAT MODEL

We consider only drive-by mining rather than legitimate browser-based mining in our threat model, i.e., we measure only the prevalence of mining without users' consent. A website may host stealthy miners for many reasons. Some website owners knowingly include

[1]Note that Monero is not the only altcoin that uses the CryptoNight algorithm: most CPU-mineable coins that exist today, such as Bytecoin, Bitsum, Masari, Stellite, AEON, Graft, Haven Protocol, Intense Coin, Loki, Electroneum, BitTube, Dero, LeviarCoin, Sumokoin, Karbo, Dinastycoin, and TurtleCoin are based on CryptoNight.

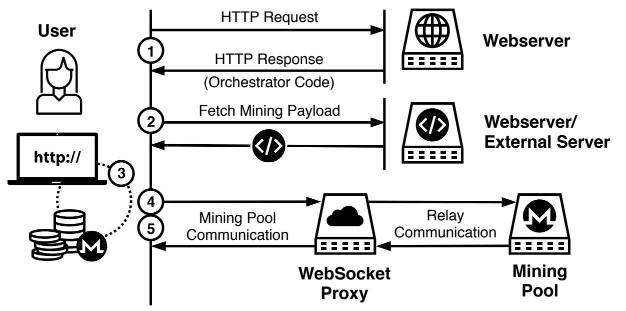

Figure 1: Overview of a typical drive-by mining attack.

them on their sites without informing the users to monetize their sites on the sly. However, it is also possible that the owners are unaware that their site is stealing CPU cycles from their visitors. For instance, silent cryptocurrency miners may ship with advertisements or third-party services. In some cases, the attackers install the miners after they compromise a victim site. In this research, we measure, analyze, and detect all these cases of drive-by mining.

Figure 1 illustrates a typical drive-by mining attack. A cryptocurrency mining script contains two components: the *orchestrator* and the *mining payload*. When a user visits a drive-by mining website, the website (1) serves the orchestrator script, which checks the host environment to find out how many CPU cores are available, (2) downloads the highly-optimized cryptomining payload (as either Wasm or asm.js) from the website or an external server, (3) instantiates a number of web workers [70], i.e., spawns separate threads, with the mining payload, depending on how many CPU cores are available, (4) sets up the connection with the mining pool server through a WebSocket proxy server, and, (5) finally, fetches work from the mining pool and submits the hashes to the mining pool through the WebSocket proxy server. The protocol used for this communication with the mining pool is usually Stratum.

4 DRIVE-BY MINING IN THE WILD

The goals of our large-scale analysis of active drive-by mining campaigns in the wild are two-fold: first, we investigate the prevalence and profitability of this threat to show that it makes economic sense for cybercriminals to invest in this type of attack—being a low effort heist with potentially high rewards. Second, we evaluate the effectiveness of current drive-by mining defenses, and show that they are insufficient against attackers who are already actively using obfuscation to evade detection. Based on our findings, we propose an obfuscation-resilient detection system for drive-by mining websites in Section 5.

As part of our analysis, we first crawl Alexa's Top 1 Million websites, log and analyze all code served by each website, monitor side effects caused by executing the code, and capture the network traffic between the visited website and any external server. Then, we proceed to detect cryptomining code in the logged data and the use of the Stratum protocol for communicating with mining pool servers in the network traffic of each website. Finally, we correlate the results from all websites to answer the following questions:

(1) How prevalent is drive-by mining in the wild?
(2) How many different drive-by mining services exist currently?
(3) Which evasion tactics do drive-by mining services employ?
(4) What is the modus operandi of different types of campaigns?
(5) How much profit do these campaigns make?
(6) Can we find common characteristics across different drive-by mining services that we can use for their detection?

Table 1: Summary of our dataset and key findings.

Crawling period	March 12, 2018 – March 19, 2018
# of crawled websites	991,513
# of drive-by mining websites	1,735 (0.18%)
# of drive-by mining services	28
# of drive-by mining campaigns	20
# of websites in biggest campaign	139
Estimated overall profit	US$ 188,878.84
Most profitable/biggest campaign	US$ 31,060.80
Most profitable website	US$ 17,166.97

Table 1 summarizes our dataset and key findings. We start by discussing our data collection approach in Section 4.1, explain how we identify drive-by mining websites in Section 4.2, explore websites and campaigns in-depth, as well as estimate their profit in Section 4.3, and finally summarize characteristics that are common across the identified drive-by mining services in Section 4.4.

4.1 Data Collection

As the basis for our analysis, we built a web crawler for visiting Alexa's Top 1 Million websites and collecting data related to drive-by mining. During our preliminary analysis, we observed that many malicious websites serve a mining payload only when the user visits an internal webpage. Thus, in contrast to related studies [45, 51, 57] that based their analysis only on the websites' landing pages,[2] we configured the crawler to visit three random internal pages of each website. The crawler stayed for four seconds on each visited page. Moreover, we configured it to passively collect data from each visited website without simulating any user interactions. That is, the crawler did not give any consent for cryptomining.

4.1.1 Cryptomining Code. To identify the cryptomining payloads that the drive-by mining website serves to client browsers, the web crawler saves the webpage, any embedded JavaScript, and all the requests originating from and responses served to the webpage. Then, our offline analyzer parses these logs to identify known drive-by mining services (such as Coinhive or Mineralt). As a first approximation, it does so using string matches, similar to existing defenses (see Section 2.4). However, this is only the first step in our analysis: as we show later, relying on pattern matching alone to detect drive-by mining easily leads to false negatives.

As explained in the previous section, the mining code consists of two components: the orchestrator and the optimized hash generation code (i.e., the mining payload), which we can both identify independently of each other.

Identification of the orchestrator. Usually, websites embed the orchestrator script in the main webpage, which we can detect by looking for specific string patterns. For instance, Listing 1 shows

[2]PublicWWW [12] only recently started indexing internal pages: `https://twitter.com/bad_packets/status/1029553374897696768` (August 14, 2018)

Table 2: Types of mining services in our initial dataset and their keywords.

Mining Service	Keywords
Coinhive	new CoinHive\\.Anonymous | coinhive.com/lib/coinhive.min.js | authedmine.com/lib/
CryptoNoter	minercry.pt/processor.js | \\.User\\(addr
NFWebMiner	new NFMiner | nfwebminer.com/lib/
JSECoin	load.jsecoin.com/load
Webmine	webmine.cz/miner
CryptoLoot	CRLT\\.anonymous | webmine.pro/lib/crlt.js
CoinImp	www.coinimp.com/scripts | new CoinImp.Anonymous | new Client.Anonymous | freecontent.stream | freecontent.data | freecontent.date
DeepMiner	new deepMiner.Anonymous | deepMiner.js
Monerise	apin.monerise.com | monerise_builder
Coinhave	minescripts\\.info'
Cpufun	snipli.com/[A-Za-z]+\\" data-id='
Minr	abc\\.pema\\.cl | metrika\\.ron\\.si | cdn\\.rove\\.cl | host\\.dns\\.ga | static\\.hk\\.rs | hallaert\\.online | st\\.kjli\\.fi | minr\\.pw | cnt\\.statistic\\.date | cdn\\.static-cnt\\.bid | ad\\.g-content\\.bid | cdn\\.jquery-uim\\.download'
Mineralt	ecart\\.html\\?bdata= | /amo\\.js\\"> | mepirtedic\\.com'

Listing 1: Example usage of the Coinhive mining service.

```
<script src="https://coinhive.com/lib/coinhive.min.js">
</script>
<script>
    var miner = new CoinHive.Anonymous('CLIENT-ID',
                                        {throttle: 0.9});
    miner.start();
</script>
```

a website using Coinhive's service for drive-by mining by including the orchestrator component (`coinhive.min.js`) inside the `<script>` HTML tag. In this case, searching for keywords such as `CoinHive.Anonymous` or `coinhive.min.js` is enough to identify whether a website is using this particular drive-by mining service. We manually collected keywords for 13 well-known mining services (see Table 2) to identify the websites that are using them.

Identification of the mining payload. The orchestrator first checks whether the browser supports Wasm. If not, the browser loads the optimized hash generation mining payload in the web worker using asm.js, otherwise, the mining payload (Wasm module) is served to the client in one of the following three ways: (i) the code is stored in the orchestrator script in a text format, which is compiled at run time to create the Wasm module, (ii) the orchestrator script retrieves a pre-compiled Wasm module at run time from an external server, or (iii) the web worker itself directly downloads a compiled Wasm module from an external server and executes it. For all three cases, we could have used the Chrome browser (which supports Wasm) with the `--dump-wasm-module` flag to dump the Wasm module that the JIT engine (V8) executes. However, this flag is not officially documented [66] and at the time of our large-scale analysis we were not aware of this feature. Hence, we detect the Wasm-based mining payload in the following way: First, we dump all the JavaScript code and search for keywords, such as `cryptonight_hash` and `CryptonightWasmWrapper`; the existence of these keywords in the JavaScript implies the mining payload is served in text format. We detect the second and third way of serving the payload by logging and analyzing all the network requests and responsens from and to the browser's web worker.

Code obfuscation. We noticed that many drive-by mining services obfuscate both the strings used in the orchestrator script and in the Wasm module to defeat such keyword-based detection. Hence, we also look for other indicators for cryptomining and store the Wasm module for further analysis. In this way, we can estimate the number of drive-by mining services that employ code obfuscation during our in-depth analysis in Section 4.3.3.

4.1.2 CPU Load as a Side Effect. A cryptominer is a CPU-intensive program; hence, execution of the mining payload usually results in a high CPU load. However, websites may also intentionally throttle their CPU usage, either to evade detection or an attempt to conserve a visitor's resources. As part of our analysis, we investigate how many websites keep the CPU usage lower than a certain threshold. To this end, we configured the web crawler to log the CPU usage of each core and aggregate the usage across cores.

4.1.3 Mining Pool Communication. Typically, a miner talks to a mining pool to fetch the block's headers to start computing hashes. Stratum is the most commonly used protocol to authenticate with the mining pool or the proxy server to receive the job that needs to be solved, and, if the correct hash is computed, to announce the result. Most drive-by mining websites use WebSockets for this type of communication. As processes running in a browser sandbox are not permitted to open system sockets, WebSockets were designed to allow full-duplex, asynchronous communication between code running on a webpage and servers. As a result of using WebSockets, the operators of drive-by mining services need to set up WebSocket servers to listen for connections from their miners, and either process this data themselves if they also operate their own mining pool or *unwrap* the traffic and forward it to a public pool.

Consequently, we log all the WebSocket frames which are sent and received by the browser, as well as the AJAX request/response from the webpage. Then, we analyze the logged data to detect any mining pool communication by searching for command and keywords that are used by the Stratum protocol (listed in Table 3). During this analysis, we also observed that some websites are obfuscating the communication with the mining pool to evade detection. Thus, if the logged data does not include any text but only binary content, we mark the WebSocket communication as *obfuscated.*

Table 3: Stratum protocol commands and their keywords.

Command	Keywords
Authentication	type:auth | command:connect | identifier:handshake | command:info
Authentication accepted	type:authed | command:work
Fetch job	identifier:job | type:job | command:work | command:get_job | command:set_job
Submit solved hash	type:submit | command:share
Solution accepted	command:accepted
Set CPU limits	command:set_cpu_load

Extraction of pools, proxies and site keys. The communication between a cryptominer and the proxy server contains two interesting pieces of information: the proxy server address and the client identifier (also known as the *site key*). We also found several drive-by mining services that include the public mining pool and associated cryptocurrency wallet address that the proxy should use.

Clustering miners based on the proxy to which they connect gives us insights on the number of different drive-by mining services that are currently active. Additionally, clustering miners based on their site key can be used to identify campaigns. Finally, we can leverage information from public mining pool to estimate the profitability of different campaigns.

We extract this information by looking for keywords in each request sent from the cryptominer and its response. Table 3 lists the keywords commonly associated with each request/response pair in the Stratum protocol. For instance, if the request sent from the miner contains keywords related to *authentication*, we extract the site key from it.

4.1.4 Deployment and Dataset. We deployed our web crawler in Docker containers running on Kubernetes in an unfiltered network. We ran 50 Docker containers in parallel for one week mid-March 2018 to collect data from Alexa's Top 1 Million websites (as of February 28, 2018). Around 1% of the websites were offline or not responding, and we managed to crawl 991,513 of them. This process resulted in a total of 4.6 TB raw data, and a 550 MB database for the extracted information on identified miners, CPU load, and mining pool communication.

4.2 Data Analysis and Correlation

We first analyze the different artifacts produced by the data collection individually, i.e., the cryptomining code itself, the CPU load as a side effect, and the mining pool communication. We discuss how relying on each of these artifacts alone can lead to both false positives and false negatives, and therefore correlate our results across all three dimensions.

4.2.1 Cryptomining Code. We identified 13 well-known cryptomining services using the keywords listed in Table 2 and present our results in Table 4. We detected 866 websites (0.09%) that are using these 13 services without obfuscating the orchestrator code in the webpage. The majority of websites (59.35%) is using the Coinhive cryptomining service. We also found 65 websites using multiple cryptomining services.

We revisited this analysis after our data correlation (described in 4.2.4) and manually analysed part of the mining payloads of websites that we detected based on other signals. In this way, we extended our initial list of keywords for detecting unobfuscated payloads with **`hash_cn`**, **`cryptonight`**, **`WASMWrapper`**, and **`crytenight`**, and we were able to identify mining services that were not part of our initial dataset, but that are using CryptoNight-based payloads. In total, we could identify 1,627 websites based on either keywords in the orchestrator *or* in the mining payload.

Table 4: Distribution of well-known cryptomining services.

Mining Service	Number of Websites	Percentage
Coinhive	514	59.35%
CoinImp	94	10.85%
Mineralt	90	10.39%
JSECoin	50	5.77%
CryptoLoot	39	4.50%
CryptoNoter	31	3.58%
Coinhave	14	1.62%
Minr	13	1.50%
Webmine	8	0.92%
DeepMiner	5	0.58%
Cpufun	4	0.46%
Monerise	2	0.23%
NF WebMiner	2	0.23%
Total	866	100%

However, similar to current blacklist-based approaches, keyword-based analysis alone suffers from false positives and false negatives. In terms of false positives, this approach does not consider user consent, i.e., whether a website waits for a user's consent before executing the mining code. In terms of false negatives, this approach cannot detect drive-by mining websites that use code obfuscation and URL randomization, which we detected being applied in some form or another by 82.14% of the services in our dataset (see Section 4.3.3).

4.2.2 CPU Load as a Side Effect. Even though we logged the CPU load for each website during our crawl, we ultimately do not use these measurements to detect drive-by mining websites for the following reasons: First, since we were running the experiments in Docker containers, the other processes running on the same machine could affect and artificially inflate our CPU load measurement. Second, the crawler spends only four seconds on each webpage, thus, the page loading itself might lead to higher CPU loads.

We can, however, use these measurements to specifically look for drive-by mining websites with low CPU usage to give a lower bound for the pervasiveness of CPU throttling across miners and the false negatives that a detection approach solely relying on high CPU loads would cause.

4.2.3 Mining Pool Communication. Overall, 59,319 (5.39%) out of Alexa's Top 1 Million websites use WebSockets to communicate with external servers. Out of these, we identified 1,008 websites that are communicating with mining pool servers using the Stratum protocol based on the keywords shown in Table 3. We also found that 2,377 websites are encoding the data (as Hex code or salted Base64) that they send and receive through the WebSocket, in which case we could not determine whether they are mining cryptocurrency.

Even though we successfully identified 1,008 drive-by mining websites using this method, this detection method suffers from the following two drawbacks, causing false negatives: drive-by mining services may use a custom communication protocol (that is, different keywords than the ones presented in Table 3), or they may be obfuscating their communication with the mining pool.

4.2.4 Data Correlation. In our preliminary analysis based on keyword search, we identified 866 websites using 13 well-known cryptomining services. To determine how many of these websites start mining without waiting for a user to give her consent, for example by clicking a button (which our web crawler was not equipped to do), we leverage the identification of the Stratum protocol: we identify 402 websites, based on both their cryptomining code and the communication with external pool servers, that initiate the mining process without requiring a user's input. The remaining 464 websites either wait for the user's consent, circumvent our Stratum protocol detection, or did not initiate the Stratum communication within the timeframe our web crawler spent on the website.

To extend our detection to miners that evade keyword-based detection, we combine the collected information from the following sources:

- *Mining payload*: Websites identified based on keywords found in the mining payload.
- *Orchestrator*: Websites identified based on keywords found in the orchestrator code.
- *Stratum*: Websites identified as using the Stratum communication protocol.
- *WebSocket communication*: Websites that potentially use an obfuscated communication protocol.
- *Number of web workers*: All the in-browser cryptominers use web worker threads to generate hashes, while only 1.6% of all websites in our dataset use more than two web worker threads.

We identify drive-by mining websites by taking the *union* of all websites for which we identified the mining payload, orchestrator, *or* the Stratum protocol. We further add websites for which we identified WebSocket communication with an external server *and* more than two web worker threads.

As a result, we identify 1,735 websites as mining cryptocurrency, out of which 1,627 (93.78%) could be identified based on keywords in the cryptomining code, 1,008 (58.10%) use the Stratum protocol in plaintext, 174 (10.03%) obfuscate the communication protocol, and all the websites (100.00%) use Wasm for the cryptomining payload and open a WebSocket. Furthermore, at least 197 (11.36%) websites throttle their CPU usage to less than 50%, while for only 12 (0.69%) mining websites we observed a CPU load of less than 25%. In other words, relying on high CPU loads (e.g., $\geq$50%) for detection would result in 11.36% false negatives in this case (in addition to potentially causing false positives for other CPU-intensive loads, such as games and video codecs). Similarly, relying only on pattern matching on the payload would result in 6.23% false negatives.

Finally, in addition to the 13 well-known drive-by mining services that we started our analysis with (see Table 4), we also discovered 15 new drive-by mining services (see Section 4.3.6), for a total of 28 drive-by mining services in our dataset.

4.3 In-depth Analysis and Results

Based on the drive-by mining websites we detected during our data correlation, we now answer the questions posed at the beginning of this section.

4.3.1 User Notification and Consent. We consider cryptomining as abuse unless a user explicitly consents, e.g., by clicking a button. While one of the first court cases on in-browser mining suggests a more lenient definition of consent and only requires websites to provide a clear notification about the mining behavior to the user [33], we find that very few websites in our dataset do so.

To locate any notifications, we searched for mining-related keywords (such as `CPU`, `XMR`, `Coinhive`, `Crypto` and `Monero`) in the identified drive-by mining website's HTML content. In this way, we identified 67 out of 1,735 (3.86%) websites that inform their users about their use of cryptomining. These websites include 51 proxy servers to the Pirate Bay, as well as 16 unrelated websites, which, in some cases, justify the use of cryptomining as an alternative to advertisements.[3] We acknowledge that our findings only represent a lower bound of websites that notify their users, as the notifications could also be stored in other formats, for example as images, or be part of a website's terms of service. However, locating and parsing these terms is out of scope for this work.

We also found a number of websites that include Coinhive's AuthedMine [6] in addition to drive-by mining. AuthedMine is not part of our threat model, as it requires user opt-in, and as such, we did not include websites using it in our analysis. Still, at least four websites (based on a simple string search) include the `authedmine.min.js` script, while starting to mine right away with a separate mining script that does not require user input: three of these websites include the miners on the *same* page, while the fourth (`cnhv.co`, a proxy to Coinhive), includes AuthedMine on the landing page, and a non-interactive miner on an internal page.

4.3.2 Mining from Internal Pages. We found 744 out of 1,735 websites (42.88%) stealing the visitor's computational power only when she visits one of their internal pages, validating our decision to not only crawl the landing page of a website but also some internal pages. From the manual analysis of these websites, we found that most of them are video streaming websites: the websites start cryptomining when the visitor starts watching a video by clicking the links displayed on the landing page.

4.3.3 Evasion Techniques. We have identified three evasion techniques, which are widely used by the drive-by mining services in our dataset.

Code obfuscation. For each of the 28 drive-by mining services in our dataset we manually analyzed some of the corresponding websites, which we identified as mining, but for which we could not find any of the keywords in their cryptomining code. In this way, we identified 23 (82.14%) of drive-by mining services using

[3]Examples: "If ads are blocked, a low percentage of your CPU's idle processing power is used to solve complex hashes, as a form of micro-payment for playing the game." (`dogeminer2.com`) and "This website uses some of your CPU resources to mine cryptocurrency in favor of the website owner. This is a some [sic] sort of donation to thank the website owner for the work done, as well as to reduce the amount of advertising on the website." (`crypticrock.com`)

one or more of the following obfuscation techniques in at least one of the websites that are using them:

- *Packed code*: The compressed and encoded orchestrator script is decoded using a chain of decoding functions at run time.
- *CharCode*: The orchestrator script is converted to charCode and embedded in the webpage. At run time, it is converted back to a string and executed using JavaScript's `eval()` function.
- *Name obfuscation*: Variable names and functions names are replaced with random strings.
- *Dead code injection*: Random blocks of code, which are never executed, are added to the script to make reverse engineering more difficult.
- *Filename and URL randomization*: The name of the JavaScript file is randomized or the URL it is loaded from is shortened to avoid detection based on pattern matching.

We mainly found these obfuscation techniques applied to the orchestrator code and not to the mining payload. Since the performance of the cryptomining payload is crucial to maximize the profit from browser-based mining, the only obfuscation currently performed on the mining payload is name obfuscation.

Obfuscated Stratum communication. We only identified the Stratum protocol in plaintext (based on the keywords in Table 3) for 1,008 (58.10%) websites. We manually analyzed the WebSocket communication for the remaining 727 (41.90%) websites and found the following: (1) A common strategy to obfuscate the mining pool communication found in 174 (10.03%) websites is to encode the request, either as Hex code, or with salted Base64 encoding (i.e., adding a layer of encryption with the use of a pre-shared passphrase), before transmitting it through the WebSocket. (2) We could not identify any pool communication for the remaining 553 websites, either due to other encodings, or due to slow server connections, i.e., we were not able to observe any pool communication during the time our web crawler spent on a website, which could also be used by malicious websites as a tactic to evade detection by automated tools.

Anti-debugging tricks. We found 139 websites (part of a campaign targeting video streaming websites) that employ the following anti-debugging trick (see Listing 2): The code periodically checks whether the user is analyzing the code served by the webpage using developer tools. If the developer tools are open in the browser, it stops executing any further code.

4.3.4 *Private vs. Public Mining Pools.*

All the drive-by mining websites in our dataset connect to WebSocket proxy servers that listen for connections from their miners, and either process this data themselves (if they also operate their own mining pool), or *unwrap* the traffic and forward it to a public pool. That is, the proxy server could be connecting to a public mining or private mining pool. We identified 159 different WebSocket proxy servers being used by the 1,735 drive-by mining websites and only six of them are sending the public mining pool server address and the cryptocurrency wallet address (used by the pool administrator to reward the miner) associated with the website to the proxy server. These six websites use the following public mining pools: `minexmr.com`, `supportxmr.com`, `moneroocean.stream`, `xmrpool.eu`, `minemonero.pro`, and `aeon.sumominer.com`.

Listing 2: Anti-debugging trick used by 139 websites.

```
function check() {
    before = new Date().getTime();
    debugger;
    after = new Date().getTime();
    if (after-before > minimalUserResponseInMiliseconds) {
        document.write(" Dont open Developer Tools. ");
        self.location.replace('https:' +
             window.location.href.substring(window.
                        location.protocol.length));
    } else {
        before = null;
        after = null;
        delete before;
        delete after;
  }
  setTimeout(check, 100);
}
```

4.3.5 *Drive-by Mining Campaigns.*

To identify drive-by mining campaigns we rely on site keys and WebSocket proxy servers. If a campaign uses a public web mining service, the attacker uses the same site key and proxy server for all websites belonging to this campaign. If the campaign uses an attacker-controlled proxy server, the websites do not need to embed a site key, but the websites still connect to the same proxy. Hence, we use two approaches to find drive-by campaigns: First, we cluster websites that are using the same site key and proxy. We discovered 11 campaigns using this method (see Table 5). Second, we cluster the websites only based on the proxy and then manually verified websites from each cluster to see which mining code they are using and how they are including it. We identified nine campaigns using this method (see Table 6). In total, we identified 20 drive-by mining campaigns in our dataset. These campaigns include 566 websites (32.62%), for the remaining 1,169 (67.38%) websites we could not identify any connection.

We manually analyzed websites from each campaign to study their modus operandi. Based on this analysis, we classify the campaigns into the following categories based on their infection vector: miners injected through third-party services, miner injected through advertisement networks, and miners injected by compromising vulnerable websites. We also captured proxy servers to the Pirate Bay, which does not ask for users' explicit consent for mining cryptocurrency, but openly discusses this practice on its blog [54]. For each campaign, we estimate the number of visitors per month and their monthly profit (details on how we perform these estimations can be found in Section 4.3.7).

Third-party campaigns. The biggest campaigns we found target video streaming websites: we identified nine third-party services that provide media players that are embedded in other websites and which include a cryptomining script in their media player.

Video streaming websites usually present more than one link to a video, also known as *mirrors*. A click on such a link either loads the video in an embedded video player provided by the website, if it is hosting the video directly, or redirects the user to another website. We spotted suspicious requests originating from many such embedded video players which lead us to the discovery of eight third-party campaigns: `Hqq.tv`, `Estream.to`, `Streamplay.to`, `Watchers.to`, `bitvid.sx`, `Speedvid.net`, `FlashX.tv` and `Vidzi.tv` are the streaming websites that embed cryptomining

Table 5: Identified campaigns based on site keys, number of participating websites (#), and estimated profit per month.

Site Key	#	Main Pool	Type	Profit (US$)
"428347349263284"	139	`weline.info`	Third party (video)	$31,060.80
OT1CIcpkIOCO7yVMxcJiqmSWoDWOri06	53	`coinhive.com`	Torrent portals	$8,343.18
ricewithchicken	32	`datasecu.download`	Advertisement-based	$1,078.27
jscustomkey2	27	`207.246.88.253`	Third party (counter12.com)	$86.98
CryptoNoter	27	`minercry.pt`	Advertisement-based	$20.35
489djE22mdZ3[..]y4PBWLb4tc1X8ADsu	24	`datasecu.download`	Compromised websites	$142.40
first	23	`cloudflane.com`	Compromised websites	$120.02
vBaNYz4tVYKV9Q9tZlL0BPGq8rnZEl00	20	`hemnes.win`	Third party (video)	$303.14
45CQjsiBr46U[..]o2C5uo3u23p5SkMN	17	`rand.com.ru`	Compromised websites	$306.60
Tumblr	14	`count.im`	Third party	$11.31
ClmAXQqOiKXawAMBVzuc51G31uDYdJ8F	12	`coinhive.com`	Third party (night-skin.com)	$14.36

Table 6: Identified campaigns based on proxies, number of participating websites (#), and estimated profit per month.

WebSocket Proxy	#	Type	Profit (US$)
`advisorstat.space`	63	Advertisement-based	$321.71
`zenoviaexchange.com`	37	Advertisement-based	$1,516.08
`stati.bid`	20	Compromised websites	$34.94
`staticsfs.host`	20	Compromised websites	$384.91
`webmetric.loan`	17	Compromised websites	$181.32
`insdrbot.com`	7	Third party (video)	$1,689.26
`1q2w3.website`	5	Third party (video)	$2,012.90
`streamplay.to`	5	Third party (video)	$239.71
`estream.to`	4	Third party (video)	$872.72

scripts through embedded video players. The biggest campaign in our dataset is *Hqq player*, which we found on 139 websites through the proxy `weline.info`. We estimate that around 2,500 streaming websites are including the embedded video players from these eight services, attracting more than 250 million viewers per month. An independent study from AdGuard also reported similar campaigns in December 2017 [44], however, we could not find any indication that the video streaming websites they identified were still mining at the time of our analysis.

As part of third-party campaigns unrelated to video streaming, we found 14 pages on *Tumblr* under the domain `tumblr[.]com` mining cryptocurrency. The mining payload was introduced in the main page by the domain `fontapis[.]com`. We also found 39 websites were infected by using libraries provided by `counter12.com` and `night-skin.com`.

Advertisement-based campaigns. We found four advertisement-based campaign in our dataset. In this case, attackers publish advertisements that include cryptomining scripts through legitimate advertisement networks. If a user visits the infected website and a malicious advertisement is displayed, the browser starts cryptomining. The *ricewithchicken* campaign was spreading through the AOL advertising platform, which was recently also reported in an independent study by TrendMicro [41]. We also identified three campaigns spreading through the `oxcdn.com`, `zenoviaexchange.com` and `moradu.com` advertisement networks.

Compromised websites. We also identified five campaigns that exploited web application vulnerabilities to inject miner code into the compromised website. For all of these campaigns, the same orchestrator code was embedded at the bottom of the main HTML page

Table 7: Additional cryptomining services we discovered, number of websites (#) using them, and whether they provide a private proxy *and* private mining pool (✓).

Mining Service	#	Main Pool	Private?
CoinPot	43	`coinpot.co`	
NeroHut	10	`gnrdomimplementation.com`	✓
Webminerpool	13	`metamedia.host`	
CoinNebula	6	`1q2w3.website`	✓
BatMine	6	`whysoserius.club`	✓
Adless	5	`adless.io`	✓
Moneromining	5	`moneromining.online`	✓
Afminer	3	`afminer.com`	✓
AJcryptominer	4	`ajplugins.com`	✓
Crypto Webminer	4	`anisearch.ru`	
Grindcash	2	`ulnawoyyzbljc.ru`	
Mining.Best	1	`mining.best`	✓
WebXMR	1	`webxmr.com`	✓
CortaCoin	1	`cortacoin.com`	✓
JSminer	1	`jsminer.net`	✓

(and not loaded from any external libraries) in a similar fashion. Moreover, we could not find any relationship between the websites within the campaigns: they are hosted in different geographic locations and registered to different organizations. One of the campaigns was using the public mining pool server `minexmr.com`.[4] We checked the status of the wallet address on the mining pool's website and found that the wallet address had already been blacklisted for malicious activity.

Torrent portals. We found a campaign targeting 53 torrent portals, all but two of which are proxies to the Pirate Bay. We estimate that all together these websites attract 177 million users a month.

4.3.6 *Drive-by Mining Services.*
We started our analysis with 13 drive-by mining services. By analyzing the clusters based on WebSocket proxy servers, we discovered 15 more *Coinhive*-like services (see Table 7). We classify these services into two categories: the first category only provides a private proxy; however, the client can specify the mining pool address that the proxy server should use as the mining pool. *Grindcash*, *Crypto Webminer*, and *Webminerpool* belong to this category. The second category provides a private

[4]site key: *489djE22mdZ3j34vhES98tCzfVn57Wq4fA8JR6uzgHqYCfYE2nmaZxmjepwr3-GQAZd3qc3imFyGPHBy4PBWLb4tc1X8ADsu*

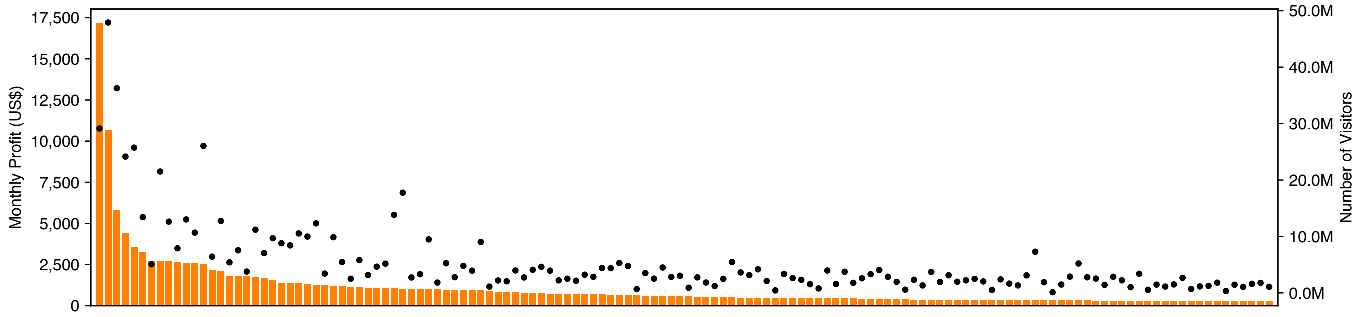

Figure 2: Profit estimation and visitor numbers for the 142 drive-by mining websites earning more than US$ 250 a month.

Table 8: Hash rate (H/s) on various mobile devices and laptops/desktops using Coinhive's in-browser miner.

	Device Type	Hash Rate (H/s)
Mobile Device	Nokia 3	5
	iPhone 5s	5
	iPhone 6	7
	Wiko View 2	8
	Motorola Moto G6	10
	Google Pixel	10
	OnePlus 3	12
	Huawei P20	13
	Huawei Mate 10 Lite	13
	iPhone 6s	13
	iPhone SE	14
	iPhone 7	19
	OnePlus 5	21
	Sony Xperia	24
	Samsung Galaxy S9 Plus	28
	iPhone 8	31
	Mean	*14.56*
Laptop Desktop	Intel Core i3-5010U	16
	Intel Core i7-6700K	65
	Mean	*40.50*

proxy and a private mining pool. The remaining services listed in Table 7 belong to this category, except for *CoinPot*, which provides a private proxy but uses Coinhive's private mining pool.

4.3.7 *Profit Estimation.*

All of the 1,735 drive-by mining websites in our dataset mine the CryptoNight-based Monero (XMR) cryptocurrency using mining pools. Almost all of them (1,729) use a site key and a WebSocket proxy server to connect to the mining pool; hence, we cannot determine their profit based on their wallet address and public mining pools.

Instead, we estimate the profit per month for all 1,735 drive-by mining websites in the following way: we first collect statistics on monthly visitors, the type of the device the visitor uses (laptop/desktop or mobile) and the time each visitor spends on each website on average from SimilarWeb [13]. We retrieved the average of these statistics for the time period from March 1, 2018 to May 31, 2018. SimilarWeb did not provide data for 30 websites in our dataset, hence, we consider only the remaining 1,705 websites.

We further need to estimate the average computing power, i.e., the hash rate per second (H/s), of each visitor. Since existing hash rate measurements [2] only consider native executables and are thus higher than the hash rates of in-browser miners—Coinhive states their Wasm-based miner achieves 65% of the performance of their native miner [5],—we performed our own measurements. Table 8 shows our results: According to our experiments, an Intel Core i3 machine (laptop) is capable of at least 16 H/s, while an Intel Core i7 machine (desktop) is capable of at least 65 H/s using the CryptoNight-based in-browser miner from Coinhive. We use their hash rates (40.50 H/s) as the representative hash rate for laptops and desktops. For the mobile devices, we calculated the mean of the hash rates (14.56 H/s) that we observed on 16 different devices. Finally, we use the API provided by MineCryptoNight [9] to calculate the mining reward in US$ for these hash rates and estimate the profit based on SimilarWeb's visitor statistics.

When looking at the profit of individual websites (see Figure 2 for the most profitable ones), we estimate that the two most profitable websites are earning US$ 17,166.97 and US$ 10,667.82 a month from 29.13 million visitors (`tumangaonline.com`, average visit of 18.12 minutes) and 47.91 million visitors (`xx1.me`, average visit of 7.45 minutes), respectively. However, there is a long tail of websites with very low profits: on average, each of the 1,705 websites earned US$ 110.77 a month, and 900, around half of the websites in our dataset, earned less than US$ 10.

Still, drive-by mining can provide a steady income stream for cybercriminals, especially when considering that many of these websites are part of campaigns. We present the results, aggregated per campaign, in Table 5 and Table 6: the most profitable campaign, spread over 139 websites, potentially earned US$ 31,060.80 a month. In total, we estimate the profit of all 20 campaigns at US$ 48,741.12. However, almost 70% of websites in our dataset were not part of any campaign, and we estimate the total profit across all websites and campaigns at US$ 188,878.85.

Note that we only estimated the profit based on the websites and campaigns captured by crawling Alexa's Top 1 Million websites, and the same campaigns could make additional profit through websites not part of this list. As a point of reference, concurrent work [57] calculated the total monthly profit of *only* the Coinhive service and *including legitimate mining*, i.e., user-approved mining through for example AuthedMine, at US$ 254,200.00 (at a market value of US$ 200) in May 2018. We base our estimations on Monero's market values on May 3, 2018 (1 XMR = US$ 253) [9]. The market value of Monero, as for any cryptocurrency, is highly volatile and fluctuated between US$ 488.80 and US$ 45.30 in the last year [7], and, thus profits may vary widely based on the current value of the currency.

4.4 Common Drive-by Mining Characteristics

Based on our analysis, we found the following common characteristics among all the identified drive-by mining services: (1) All services use CryptoNight-based cryptomining implementations. (2) All identified websites use a highly-optimized Wasm implementation of the CryptoNight algorithm to execute the mining code in the browser at native speed.[5] Moreover, our manual analysis of the Wasm implementation showed that the only obfuscation performed on Wasm modules is *name obfuscation* (all strings are stripped); any further code obfuscation applied to the Wasm module would degrade the performance (and hence negatively impact the profit). (3) All drive-by mining websites use WebSockets to communicate with the mining pool through a WebSocket proxy server.

We use our findings as the basis for MineSweeper, a detection system for Wasm-based drive-by mining websites, which we describe in the next section.

5 DRIVE-BY MINING DETECTION

Building on the findings of our large-scale analysis, we propose MineSweeper, a novel technique for drive-by mining detection, which relies neither on blacklists nor on heuristics based on CPU usage. In the arms race between defenses trying to detect the miners and miners trying to evade the defenses, one of the few gainful ways forward for the defenders is to target properties of the mining code that would be impossible or very painful for the miners to remove. The more fundamental the properties, the better.

To this end, we characterize the key properties of the hashing algorithms used by miners for specific types of cryptocurrencies. For instance, some hashing algorithms, such as CryptoNight, are *fundamentally* memory-hard. Distilling the measurable properties from these algorithms allows us to detect not just one specific variant, but *all* variants, obfuscated or not. The idea is that the only way to bypass the detector is to cripple the algorithm.

MineSweeper takes the URL of a website as the input. It then employs three approaches for the detection of Wasm-based cryptominers, one for miners using mild variations or obfuscations of CryptoNight (Section 5.3.1), one for detecting cryptographic functions in a generic way (Section 5.3.2), and one for more heavily obfuscated (and performance-crippled) code (Section 5.3.3). For the first two approaches, MineSweeper statically analyses the Wasm module used by the website, for the third one it monitors the CPU cache events during the execution of the Wasm module. During the Wasm-based analysis, MineSweeper analyses the module for the core characteristics of specific classes of the algorithm. We use a coarse but effective measure to identify cryptographic functions in general, by measuring the number of cryptographic operations (as reflected by XOR, shift, and rotate operations). We focus on the CryptoNight algorithm and its variants, since it is used by all of the cryptominers we observed so far, but it is trivial to add other algorithms.

[5]We also identified JSEminer in our dataset, which only supports asm.js; however, unlike the other services, the orchestrator code provided by this service always asks for a user's consent. For this reason, we do not classify the 50 websites using JSEminer as drive-by mining websites.

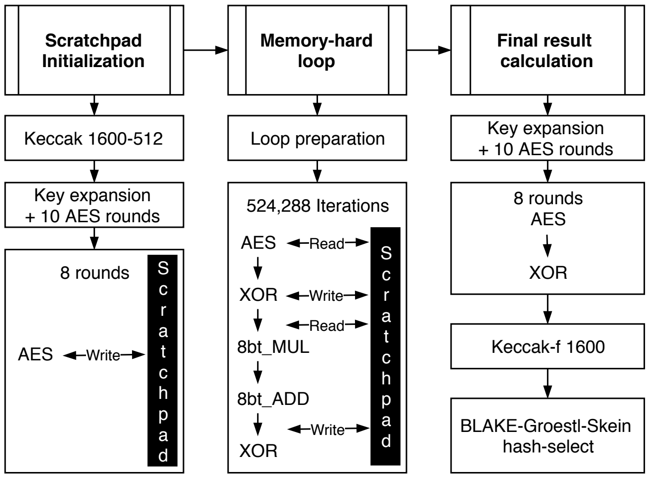

Figure 3: Components of the CryptoNight algorithm [61].

5.1 Cryptomining Hashing Code

The core component of drive-by miners, i.e., the hashing algorithm, is instantiated within the web workers responsible for solving the cryptographic puzzle. The corresponding Wasm module contains all the corresponding computationally-intensive hashing and cryptographic functions. As mentioned, all of the miners we observed mine CryptoNight-based cryptocurrencies. In this section, we discuss the key properties of this algorithm.

The original CryptoNight algorithm [61] was released in 2013 and represents, at heart, a memory-hard hashing function. The algorithm is explicitly amenable to cryptomining on ordinary CPUs, but inefficient on today's special purpose devices (ASICs). Figure 3 summarizes the three main components of the CryptoNight algorithm, which we describe below:

Scratchpad initialization. First, CryptoNight hashes the initial data with the Keccak algorithm (i.e., SHA-3), with the parameters $b = 1600$ and $c = 512$. Bytes 0–31 of the final state serve as an AES-256 key and expand to 10 round keys. Bytes 64–191 are split into 8 blocks of 16 bytes, each of which is encrypted in 10 AES rounds with the expanded keys. The result, a 128-byte block, is used to initialize a scratchpad placed in the L3 cache through several AES rounds of encryption.

Memory-hard loop. Before the main loop, two variables are created from the XORed bytes 0–31 and 32–63 of Keccak's final state. The main loop is repeated 524,288 times and consists of a sequence of cryptographic and read and write operations from and to the scratchpad.

Final result calculation. The last step begins with the expansion of bytes 32–63 from the initial Keccak's final state into an AES-256 key. Bytes 64-191 are used in a sequence of operations that consists of an XOR with 128 scratchpad bytes and an AES encryption with the expanded key. The result is hashed with Keccak-f (which stands for Keccak permutation) with $b = 1600$. The lower 2 bits of the final state are then used to select a final hashing algorithm to be applied from the following: BLAKE-256, Groestl-256, and Skein-256.

There exist two CryptoNight variants made by Sumokoin and AEON, *cryptonight-heavy* and *cryptonight-light*, respectively. The main difference between these variants and the original design is the dimension of the scratchpad: the light version uses a scratchpad size of 1 MB, and the heavy version a scratchpad size of 4 MB.

5.2 Wasm Analysis

To prepare a Wasm module for analysis, we use the WebAssembly Binary Toolkit (WABT) debugger [14] to translate it into linear assembly bytecode. We then perform the following static analysis steps on the bytecode:

Function identification. We first identify functions and create an internal representation of the code for each function. If the names of the functions are stripped, as part of common name obfuscation, we assign them an identifier with an increasing index.

Cryptographic operation count. In the second step, we inspect the identified functions one by one in order to track the appearance of each relevant Wasm operation. More precisely, we first determine the structure of the control flow by identifying the control constructs and instructions. We then look for the presence of operations commonly used in cryptographic operations (XOR, shift and rotate instructions). In many cryptographic algorithms, these operations take place in loops, so we specifically use the knowledge of the control flow to track such operations in loops. However, doing so is not always enough. For instance, at compile time, the Wasm compiler unrolls some of the loops to increase the performance. Since we aim to detect all loops, including the unrolled ones, we identify repeated flexible-length sequences of code containing cryptographic operations and mark them as a loop if a sequence is repeated for more than five times.

5.3 Cryptographic Function Detection

Based on our static analysis of the Wasm modules, we now detect the CryptoNight's hashing algorithm. We describe three approaches: one for mild variations or obfuscations of CryptoNight, one for detecting any generic cryptographic function, and one for more heavily obfuscated code.

5.3.1 Detection Based on Primitive Identification. The CryptoNight algorithm uses five cryptographic primitives, which are all necessary for correctness: Keccak (Keccak 1600-512 and Keccak-f 1600), AES, BLAKE-256, Groestl-256, and Skein-256. MineSweeper identifies whether any of these primitives are present in the Wasm module by means of fingerprinting. It is important to note that the CryptoNight algorithm and its two variants must use *all* of these primitives in order to compute a correct hash; by detecting the use of *any* of them, our approach can also detect payload implementation split across modules.

We create fingerprints of the primitives based on their specification, as well as the manual analysis of 13 different mining services (as presented in Table 2). The fingerprints essentially consist of the count of cryptographic operations in functions, and more specifically within regular and unrolled loops. We then look for the closest match of a candidate function in the bytecode to each of the primitive fingerprints based on the cryptographic operation count. To this end, we compare every function in the Wasm module one by one with the fingerprints and compute a "similarity score" of how many types of cryptographic instructions that are present in the fingerprint are also present in the function, and a "difference score" of discrepancies between the number of each of those instructions in the function and in the fingerprint. As an example, assume the fingerprint for BLAKE-256 has 80 XOR, 85 left shift, and 32 right shift instructions. Further assume, the function `foo()`, which is an implementation of BLAKE-256, that we want to match against this fingerprint, contains 86 XOR, 85 left shift, and 33 right shift instructions. In this case, the similarity score is 3, as all three types of instructions are present in `foo()`, and the difference score is 2, because `foo()` contains an extra XOR and an extra shift instruction.

Together, these scores tell us how close the function is to the fingerprint. Specifically, for a match we select the functions with the highest similarity score. If two candidates have the same similarity score, we pick the one with the lowest difference score. Based on the similarity score and difference score we calculated for each identified functions, we classify them in three categories: full match, good match, or no match. For a full match, all types of instructions from the fingerprint are also present in the function, and the difference score is 0. For a good match, we require at least 70% of the instruction types in the fingerprint to be contained in the function, and a difference score of less than three times the number of instruction types.

We then calculate the likelihood that the Wasm module contains a CryptoNight hashing function based on the number of primitives that successfully matched (either as a full or a good match). The presence of even one of these primitives can be used as an indicator for detecting potential mining payloads, but we can also set more conservative thresholds, such as flagging a Wasm module as a CryptoNight miner if only two or three out of the five cryptographic primitives are fully matched. We evaluate the number of primitives that we can match across different Wasm-based cryptominer implementations in Section 6.

5.3.2 Generic Cryptographic Function Detection. In addition to detecting the cryptographic primitives specific to the CryptoNight algorithm, our approach also detects the presence of cryptographic functions in a Wasm module in a more generic way. This is useful for detecting potential new CryptoNight variants, as well as other hashing algorithms. To this end, we count the number of cryptographic operations (XOR, shift, and rotate operations) inside loops in each function of the Wasm module, and flag a function as a cryptographic function if this number exceeds a certain threshold.

5.3.3 Detection Based on CPU Cache Events. While not yet an issue in practice, in the future, cybercriminals may well decide to sacrifice profits and highly obfuscate their cryptomining Wasm modules in order to evade detection. In that case, the previous algorithm is not sufficient. Therefore, as a last detection step, MineSweeper also attempts to detect cryptomining code by monitoring CPU cache events during the execution of a Wasm module—a fundamental property for any reasonably efficient hashing algorithm.

In particular, we make use of how CryptoNight explicitly targets mining on ordinary CPUs rather than on ASICs. To achieve this, it relies on random accesses to slow memory and emphasizes latency dependence. For efficient mining, the algorithm requires about 2 MB of fast memory per instance.

This is favorable for ordinary CPUs for the following reasons [61]:

(1) Evidently, 2 MB do not fit in the L1 or L2 cache of modern processors. However, they fit in the L3 cache.
(2) 1 MB of internal memory is unacceptable for today's ASICs.
(3) Moreover, even GPUs do not help. While they may run hundreds of code instances concurrently, they are limited in their memory speeds. Specifically, their GDDR5 memory is much slower than the CPU L3 cache. Additionally, it optimizes pure bandwidth, but not random access speed.

MineSweeper uses this fundamental property of the CryptoNight algorithm to identify it based on its CPU cache usage. Monitoring L1 and L3 cache events using the Linux `perf` [1] tool during the execution of a Wasm module, MineSweeper looks for load and store events caused by random memory accesses. As our experiments in Section 6 demonstrate, we can observe a significantly higher load/store frequency during the execution of a cryptominer payload compared to other use cases, including video players and games, and thus detect cryptominers with high probability.

5.4 Deployment Considerations

While MineSweeper can be used for the profiling of websites as part of large-scale studies such as ours, we envision it as a tool that notifies users about a potential drive-by mining attack while browsing and gives them the option to opt-out, e.g., by not loading Wasm modules that trigger the detection of cryptographic primitives, or by suspending the execution of the Wasm module as soon as suspicious cache events are detected.

Our defense based on the identification of cryptographic primitives could be easily integrated into browsers, which, so far, mainly rely on blacklists and CPU throttling of background scripts as a last line of defense [21, 22, 29]. As our approach is based on static analysis, browsers could use our techniques to profile Wasm modules as they are loaded and ask the user for permission before executing them. As an alternative and browser-agnostic deployment strategy, SEISMIC [69] instruments Wasm modules to profile their use of cryptographic operations during execution, although this approach comes with considerable run-time overhead.

Integrating our defense based on monitoring cache events, unfortunately, is not so straightforward: access to performance counters requires root privileges and would need to be implemented by the operating system itself.

6 EVALUATION

In this section, we evaluate the effectiveness of MineSweeper's components based on static analysis of the Wasm code and CPU cache event monitoring for the detection of the cryptomining code currently used by drive-by mining websites in the wild. We further compare MineSweeper to a state-of-the-art detection approach based on blacklisting. Finally, we discuss the penalty in terms of performance, and thus profits, evasion attempts against MineSweeper would incur.

Dataset. To test our Wasm-based analysis we crawled Alexa's Top 1 Million websites a second time over the period of one week in the beginning of April 2018 with the sole purpose of collecting Wasm-based mining payloads. This time we configured the crawler to visit only the landing page of each website for a period of four seconds. The crawl successfully captured 748 Wasm modules served by 776 websites. For the remaining 28 modules, the crawler was killed before it was able to dump the Wasm module completely.

Table 9: Results of our cryptographic primitive identification. MineSweeper detected at least two of CryptoNight's primitives in *all* mining samples with no false positives.

Detected Primitives	Number of Wasm Samples	Number of Cryptominers	Missing Primitives
5	30	30	-
4	3	3	AES
3	-	-	-
2	3	3	Skein, Keccak, AES
1	-	-	-
0	4	0	All

Evaluation of cryptographic primitive identification. Even though we were able to collect 748 valid Wasm modules, only 40 among them are, in fact, unique. This is because many websites use the same cryptomining services. We also found that some of these cryptomining services are providing different versions of their mining payload. Table 9 shows our results for the CryptoNight function detection on these 40 unique Wasm samples. We were able to identify all five cryptographic primitives of CryptoNight in 30 samples, four primitives in three samples and two primitives in another three samples. In these last three samples, we could only detect the Groestl and BLAKE primitives, which suggests that these are the most reliable primitives for this detection. As part of an in-depth analysis, we identified these samples as being part of the mining services BatMine and Webminerpool (two of the samples are a different version of the latter), which were not part of our dataset of mining services that we used for the fingerprint generation, but rather services we discovered during our large-scale analysis.

However, our approach did not produce any false positives and the four samples in which MineSweeper did not detect any cryptographic primitive were, in fact, benign: an online magazine reader, a videoplayer, a node library to represent a 64-bit two's-complement integer value, and a library for hyphenation. Furthermore, the generic cryptographic function detection successfully flagged all 36 mining samples as positives and all four benign cases as negatives.

Evaluation of CPU cache event monitoring. For this evaluation we used `perf` to capture L1 and L3 cache events when executing various types of web applications. We conducted all experiments on an Intel Core i7-930 machine running Ubuntu 16.04 (baseline). We captured the number of L1 data cache loads, L1 data cache stores, L3 cache stores, and L3 cache loads within 10 seconds when visiting four categories of web applications: cryptominers (Coinhive and NFWebMiner, both with 100% CPU usage), video players, Wasm-based games, and JavaScript (JS) games. We visited seven websites from each category and calculated the mean and standard deviation (stdev) of all the measurements for each category.

As Figure 4 (left) and Figure 5 (left) show, that L1 and L3 cache events are very high for the web applications that are mining cryptocurrency, but considerably lower for the other types of web applications. Compared to the second most cache-intensive applications,

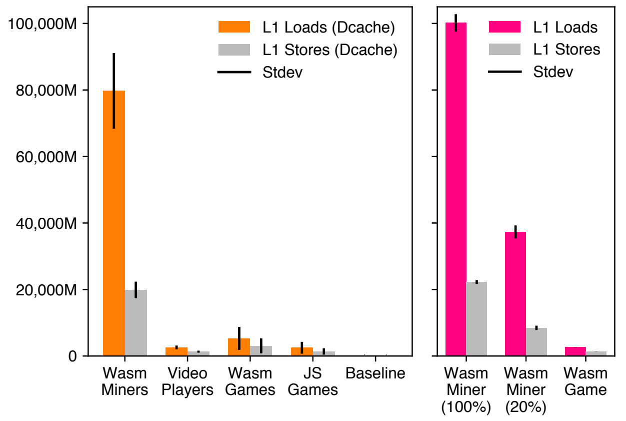

Figure 4: Performance counter measurements for the L1 data cache for miners and other web applications on two different machines (# of operations per 10 seconds, M=million).

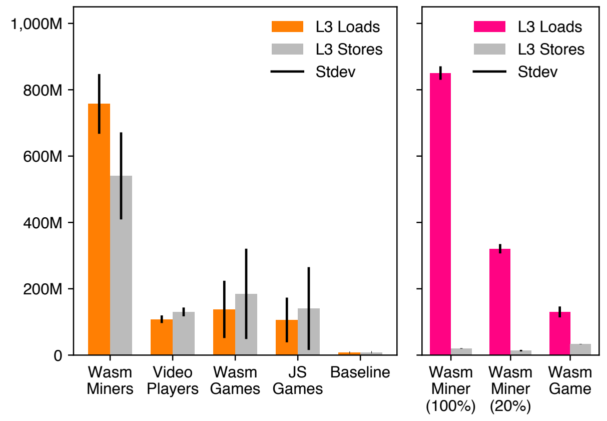

Figure 5: Performance counter measurements for the L3 cache for miners and other web applications on two different machines (# of operations per 10 seconds, M=million).

Wasm-based games, the Wasm-based miners perform on average 15.05*x* as many L1 data cache loads, and 6.55*x* as many L1 data cache stores. The difference for the L3 cache is less severe, but still noticeable: here on average the miners perform 5.50*x* and 2.93*x* as many cache loads and stores, respectively, compared to the games.

We performed a second round of experiments on a different machine (Intel Core i7-6700K), which has a slightly different cache architecture, to verify the reliability of the CPU cache events. We also used these experiments to investigate the effect of CPU throttling on the number of cache events. Coinhive's Wasm-based miner allows throttling in increments of 10% intervals. We configured it to use 100% CPU and 20% CPU and compared it against a Wasm-based game. We executed the experiments 20 times and calculated the mean and standard deviation (stdev). As Figure 4 (right) and Figure 5 (right) show, on this machine L3 cache store events cannot be used for the detection of miners: we observed only a low number of L3 cache stores overall, and on average more stores for the game than for the miners. However, L3 cache loads, as well as L1 data cache loads and stores are a reliable indicator for mining. When using only 20% of the CPU, we still observed 37.25%, 38.05%, and 37.71% of the average number of events compared to 100% CPU usage for L1 data cache loads, L1 data cache stores, and L3 cache loads, respectively. Compared to the game, the miner performed 13.96*x* and 6.29*x* as many L1 data cache loads and stores, and 2.46*x* as many L3 cache loads even when utilizing only 20% of the CPU.

Comparison to blacklisting approaches. To compare our approach against existing blacklisting-based defenses we evaluate MineSweeper against Dr. Mine [8]. Dr. Mine uses CoinBlockerLists [4] as the basis to detect mining websites. For the comparison we visited the 1,735 websites that were mining during our first crawl for the large-scale analysis in mid-March 2018 (see Section 4) with both tools. We made sure to use updated CoinBlockerLists and executed Dr. Mine and MineSweeper in parallel to maximize the chance that the same drive-by mining websites would be active. During this evaluation, on May 9, 2018, Dr. Mine could only find 272 websites, while MineSweeper found 785 websites that were still actively mining cryptocurrency. Furthermore, all the 272 websites identified by Dr. Mine are also identified by MineSweeper.

Impact of evasion techniques. In order to evade our identification of cryptographic primitives, attackers could heavily obfuscate their code, or implement the CryptoNight functions completely in asm.js or JavaScript. In both cases, MineSweeper would still be able to detect the cryptomining based on the CPU cache event monitoring. To evade this type of defense, and since we are only monitoring unusually high cache load and stores that are typical for cryptomining payloads, attackers would need to slow down their hash rate, for example by interleaving their code with additional computations that have no effect on the monitored performance counters.

In the following, we discuss the performance hit (and thus loss of profit) that alternative implementations of the mining code in asm.js, and an intentional sacrifice of the hash rate, in this case by throttling the CPU usage, would incur. Table 10 show our estimation for the potential performance and profit losses on a high-end (Intel Core i7-6700K) and a low-end (Intel Core i3-5010U) machine. As an illustrative example, we assume that in the best case an attacker is able to make a profit of US$ 100 with the maximum hash rate of 65 H/s on the i7 machine. Just falling back to asm.js would cost an attacker 40.00%–43.75% of her profits (with a CPU usage of 100%). Moreover, throttling the CPU speed to 25% on top of falling back to asm.js would cost her 85.00%–85.94% of her profits, leaving her with only US$ 15.00 on a high-end and US$ 3.46 on a low-end machine. In more concrete numbers from our large-scale analysis of drive-by mining campaigns in the wild (see Section 4.3), the most profitable campaign, which is potentially earning US$ 31,060.80 a month (see Table 5), would only earn US$ 4,367.15 a month.

7 LIMITATIONS AND FUTURE WORK

Our large-scale analysis of drive-by mining in the wild likely missed active cryptomining websites due to limitations of our crawler. We only spend four seconds on each webpage; hence we could have missed websites that wait for a certain amount of time before serving the mining payload. Similarly, we are not able to capture the mining pool communication for websites that implement mining delays, and in some cases due to slow server connections, which exceed the timeout of our crawler. Moreover, we only visit each webpage once, but, some cryptomining payloads, especially the

Table 10: Decrease in the hash rate (H/s), and thus profit, *compared to the best-case scenario* (∗) using Wasm with 100% CPU utilization if asm.js is being used and the CPU is throttled on an Intel Core i7-6700K and an Intel Core i3-5010U machine.

	Baseline		100% CPU			75% CPU					50% CPU					25% CPU				
	H/s Wasm	Profit US$	H/s asm.js	H/s Loss	Profit US$	H/s Wasm	Profit US$	H/s asm.js	H/s Loss	Profit US$	H/s Wasm	Profit US$	H/s asm.js	H/s Loss	Profit US$	H/s Wasm	Profit US$	H/s asm.js	H/s Loss	Profit US$
i7	*65*∗	$100.00	39	40.00%	$60.00	48.75	$75.00	29.25	55.00%	$45.00	32.5	$50.00	19.5	70.00%	$30.00	16.25	$25.00	9.75	85.00%	$15.00
i3	*16*∗	$24.62	9	43.75%	$13.85	12	$18.46	6.75	57.81%	$10.38	8	$12.31	4.5	71.88%	$6.92	4	$6.15	2.25	85.94%	$3.46

ones that spread through advertisement networks, are not served on every visit. Our crawler also did not capture the cases in which cryptominers are loaded as part of "pop-under" windows. Furthermore, the crawler visited each website with the User Agent String of the Chrome browser on a standard desktop PC. We leave the study of campaigns specifically targeting other devices, such as Android phones, for future work. Another avenue for future work is studying the longevity of the identified campaigns. We based our profit estimations on the assumption that they stayed active for at least a month, but they might have been disrupted earlier.

Our defense based on static analysis is similarly prone to obfuscation as any related static analysis approach. However, even if attackers decide to sacrifice performance (and profits) for evading our defense through obfuscation of the cryptomining payload, we would still be able to detect the mining based on monitoring the CPU cache. Trying to evade this detection technique by adding additional computations would severely degrade the mining performance—to a point that it is not profitable anymore.

Furthermore, currently all drive-by mining services use Wasm-based cryptomining code, and hence, we implemented our defense only for this type of payload. Nevertheless, we could implement our approach also for the analysis of asm.js in future work. Finally, our defense is tailored for detecting cryptocurrencies using the CryptoNight algorithm, as these are currently the only cryptocurrencies that can profitably be mined using regular CPUs [9]. Even though our generic cryptographic function detection did not produce any false positives in our evaluation, we still can imagine many benign Wasm modules using cryptographic functions for other purposes. However, Wasm is not widely adopted yet for other use cases besides drive-by mining and we therefore could not evaluate our approach on a larger dataset of benign applications.

8 RELATED WORK

Related work has extensively studied how and why attackers compromise websites through the exploitation of software vulnerabilities [16, 18], misconfigurations [23], inclusion of third-party scripts [48], and advertisements [75]. Traditionally, the attackers' goals ranged from website defacements [17, 42], over enlisting the website's visitors into distributed denial-of-service (DDoS) attacks [53], to the installation of exploit kits for drive-by download attacks [30, 55, 56], which infect visitors with malicious executables. In comparison, the abuse of the visitors' resources for cryptomining is a relatively new trend.

Previous work on cryptomining focused on botnets that were used to mine Bitcoin during the year 2011–2013 [34]. The authors found that while mining is less profitable than other malicious activities, such as spamming or click fraud, it is attractive as a secondary monetizing scheme, as it does not interfere with other revenue-generating activities. In contrast, we focused our analysis on drive-by mining attacks, which serve the cryptomining payload as part of infected websites, and not malicious executables. The first other study in this direction was recently performed by Eskandari et al. [25]. However, they based their analysis solely on looking for the `coinhive.min.js` script within the body of each website indexed by Zmap and PublicWWW [45]. In this way, they were only able to identify the Coinhive service. Furthermore, contrary to the observations made in their study, we found that attackers have found valuable targets, such as online video streaming, to maximize the time users spend online, and consequently the revenue earned from drive-by mining. Concurrently to our work, Papadopoulos et al. [51] compared the potential profits from drive-by mining to advertisement revenue by checking websites indexed by PublicWWW against blacklists from popular browser extensions. They concluded that mining is only more profitable than advertisements when users stay on a website for longer periods of time. In another concurrent work, Rüth et al. [57] studied the prevalence of drive-by miners in Alexa's Top 1 Million websites based on JavaScript code patterns from a blacklist, as well as based on signatures generated from SHA-255 hashes of the Wasm code's functions. They further calculated the Coinhive's overall monthly profit, which includes legitimate mining as well. In contrast, we focus on the profit of individual campaigns that perform mining without their user's explicit consent. Furthermore, with MineSweeper, we also present a defense against drive-by mining that could replace current blacklisting-based approaches.

The first part of our defense, which is based on the identification of cryptographic primitives is inspired by related work on identifying cryptographic functionality in desktop malware, which frequently uses encryption to evade detection and secure the communication with its command-and-control servers. Gröbert et al. [31] attempt to identify cryptographic code and extract keys based on dynamic analysis. Aligot [38] identifies cryptographic functions based on their input-output (I/O) characteristics. Most recently, CryptoHunt [72] proposed to use symbolic execution to find cryptographic functions in obfuscated binaries. In contrast to the heavy use of obfuscation in binary malware, obfuscation of the cryptographic functions in drive-by miners is much less favorable for attackers. Should they start to sacrifice profits in favor of evading defenses in the future, we can explore the aforementioned more sophisticated detection techniques for detecting cryptomining code. For the time being, relatively simple fingerprints of instructions that are commonly used by cryptographic operations are enough to reliably detect cryptomining payloads, as also observed by Wang et al. [69] in concurrent work. Their approach, SEISMIC, generates signatures based on counting the execution of five arithmetic instructions that are commonly used by Wasm-based miners. In contrast to profiling whole Wasm modules, we detect the individual cryptographic primitives of the cryptominers' hashing algorithms, and also supplement our approach by looking for suspicious memory access patterns.

This second part of our defense, which is based on monitoring CPU cache events, is related to CloudRadar [76], which uses performance counters to detect the execution of cryptographic applications and to defend against cache-based side-channel attacks in the cloud. Finally, the most closely related work in this regard is MineGuard [64], also a hypervisor tool, which uses signatures bases on performance counters to detect both CPU- and GPU-based mining executables on cloud platforms. Similar to our work, the authors argue that the evasion of this type of detection would make mining unprofitable—or at least less of a nuisance to cloud operators and users by consuming fewer resources.

9 CONCLUSION

In this paper, we examined the phenomenon of drive-by mining. The rise of mineable alternative coins (altcoins) and the performance boost provided to in-browser scripting code by WebAssembly, have made such activities quite profitable to cybercriminals: rather than being a one-time heist, this type of attack provides continuous income to an attacker.

Detecting miners by means of blacklists, string patterns, or CPU utilization alone is an ineffective strategy, because of both false positives and false negatives. Already, drive-by mining solutions are actively using obfuscation to evade detection. Instead of the current inadequate measures, we proposed MineSweeper, a new detection technique tailored to the algorithms that are fundamental to the drive-by mining operations—the cryptographic computations required to produce valid hashes for transactions.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and input to improve the paper. We also thank Kevin Borgolte, Aravind Machiry, and Dipanjan Das for supporting the cloud infrastructure for our experiments.

This research was supported by the MALPAY consortium, consisting of the Dutch national police, ING, ABN AMRO, Rabobank, Fox-IT, and TNO. This paper represents the position of the authors and not that of the aforementioned consortium partners. This project further received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 786669. Any dissemination of results must indicate that it reflects only the authors' view and that the Agency is not responsible for any use that may be made of the information it contains.

This material is also based upon research sponsored by DARPA under agreement number FA8750-15-2-0084, by the ONR under Award No. N00014-17-1-2897, by the NSF under Award No. CNS-1704253, SBA Research, and a Security, Privacy and Anti-Abuse award from Google. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, by our sponsors.

REFERENCES

[1] perf: Linux profiling with performance counters. `https://perf.wiki.kernel.org/index.php/Main_Page` (2015).
[2] CPU for Monero. `https://cryptomining24.net/cpu-for-monero` (2017). (Last accessed: 2018-08-17).
[3] Alexa. `https://www.alexa.com` (2018). (Last accessed: 2018-02-28).
[4] CoinBlockerLists. `https://zerodot1.gitlab.io/CoinBlockerListsWeb` (2018). (Last accessed: 2018-05-09).
[5] Coinhive. `https://coinhive.com` (2018).
[6] Coinhive AuthedMine - A Non-Adblocked Miner. `https://coinhive.com/documentation/authedmine` (2018).
[7] CryptoCompare. `https://www.cryptocompare.com/coins/xmr/` (2018). (Last accessed: 2018-08-17).
[8] Dr. Mine. `https://github.com/1lastBr3ath/drmine` (2018).
[9] MineCryptoNight. `https://minecryptonight.net` (2018). (Last accessed: 2018-05-03).
[10] MinerBlock. `https://github.com/xd4rker/MinerBlock` (2018).
[11] No Coin. `https://github.com/keraf/NoCoin` (2018).
[12] PublicWWW. `https://publicwww.com` (2018).
[13] SimilarWeb. `https://www.similarweb.com` (2018).
[14] WABT: The WebAssembly Binary Toolkit. `https://github.com/WebAssembly/wabt` (2018).
[15] Nadav Avital, Matan Lion, and Ron Masas. Crypto Me0wing Attacks: Kitty Cashes in on Monero. `https://www.incapsula.com/blog/crypto-me0wing-attacks-kitty-cashes-in-on-monero.html` (May 2018).
[16] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. Delta: Automatic Identification of Unknown Web-based Infection Campaigns. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2013).
[17] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. Meerkat: Detecting Website Defacements through Image-based Object Recognition. In *Proc. of the USENIX Security Symposium* (2015).
[18] Davide Canali and Davide Balzarotti. Behind the Scenes of Online Attacks: an Analysis of Exploitation Behaviors on the Web. In *Proc. of the Network and Distributed System Security Symposium (NDSS)* (2013).
[19] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. I Always Feel Like Somebody's Watching Me: Measuring Online Behavioural Advertising. In *Proc. of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)* (2015).
[20] Catalin Cimpanu. Cryptojackers Found on Starbucks WiFi Network, GitHub, Pirate Streaming Sites. `https://www.bleepingcomputer.com/news/security/cryptojackers-found-on-starbucks-wifi-network-github-pirate-streaming-sites/` (December 2017).
[21] Catalin Cimpanu. Firefox Working on Protection Against In-Browser Cryptojacking Scripts. `https://www.bleepingcomputer.com/news/software/firefox-working-on-protection-against-in-browser-cryptojacking-scripts/` (March 2018).
[22] Catalin Cimpanu. Tweak to Chrome Performance Will Indirectly Stifle Cryptojacking Scripts. `https://www.bleepingcomputer.com/news/security/tweak-to-chrome-performance-will-indirectly-stifle-cryptojacking-scripts/` (February 2018).
[23] Constanze Dietrich, Katharina Krombholz, Kevin Borgolte, and Tobias Fiebig. Investigating Operators' Perspective on Security Misconfigurations. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2018).
[24] Abeer ElBahrawy, Laura Alessandretti, Anne Kandler, Romualdo Pastor-Satorras, and Andrea Baronchelli. Bitcoin ecology: Quantifying and modelling the long-term dynamics of the cryptocurrency market. arXiv:1705.05334v3 [physics.soc-ph] (November 2017).
[25] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A First Look at Browser-based Cryptojacking. In *Proc. of the IEEE Privacy and Security on the Blockchain Workshop (IEEE S&B)* (2018).
[26] Amir Feder, Neil Gandal, JT Hamrick, Tyler Moore, and Marie Vasek. The Rise and Fall of Cryptocurrencies. In *Proc. of the Workshop on the Economics of Information Security (WEIS)* (2018).
[27] Dan Goodin. Websites use your CPU to mine cryptocurrency even when you close your browser. `https://arstechnica.com/information-technology/2017/11/sneakier-more-persistent-drive-by-cryptomining-comes-to-a-browser-near-you/` (November 2017).
[28] Dan Goodin. Now even YouTube serves ads with CPU-draining cryptocurrency miners. `https://arstechnica.com/information-technology/2018/01/now-even-youtube-serves-ads-with-cpu-draining-cryptocurrency-miners/` (January 2018).
[29] Google. Chromium Issue 766068: Please consider intervention for high cpu usage js. `https://bugs.chromium.org/p/chromium/issues/detail?id=766068` (September 2017).
[30] Chris Grier, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J. Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, Antonio Nappa, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian

Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. Manufacturing Compromise: The Emergence of Exploit-as-a-service. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2012).

[31] Felix Gröbert, Carsten Willems, and Thorsten Holz. Automated Identification of Cryptographic Primitives in Binary Programs. In *Proc. of the International Symposium on Recent Advances in Intrusion Detection (RAID)* (2011).

[32] 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. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)* (2017).

[33] John J. Hoffman, Steve C. Lee, and Jeffrey S. Jacobson. New Jersey Division of Consumer Affairs Obtains Settlement with Developer of Bitcoin-Mining Software Found to Have Accessed New Jersey Computers Without Users' Knowledge or Consent. https://nj.gov/oag/newsreleases15/pr20150526b.html (May 2015).

[34] 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 *Proc. of the Network and Distributed System Security Symposium (NDSS)* (2014).

[35] Simon Kenin. Mass MikroTik Router Infection – First we cryptojack Brazil, then we take the World? https://www.trustwave.com/Resources/SpiderLabs-Blog/Mass-MikroTik-Router-Infection---First-we-cryptojack-Brazil,-then-we-take-the-World-/ (August 2018).

[36] Brian Krebs. Who and What Is CoinHive? https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/ (March 2018).

[37] McAfee Labs. McAfee Labs Threats Report. https://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2014.pdf (June 2014).

[38] Pierre Lestringant, Frédéric Guihéry, and Pierre-Alain Fouque. Aligot: Cryptographic Function Identification in Obfuscated Binary Programs. In *Proc. of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2015).

[39] Shannon Liao. Showtime websites secretly mined user CPU for cryptocurrency. https://www.theverge.com/2017/9/26/16367620/showtime-cpu-cryptocurrency-monero-coinhive/ (September 2017).

[40] Shannon Liao. UNICEF wants you to mine cryptocurrency for charity. https://www.theverge.com/2018/4/30/17303624/unicef-mining-cryptocurrency-charity-monero/ (April 2018).

[41] Chaoying Liu and Joseph C. Chen. Cryptocurrency Web Miner Script Injected into AOL Advertising Platform. https://blog.trendmicro.com/trendlabs-security-intelligence/cryptocurrency-web-miner-script-injected-into-aol-advertising-platform/ (April 2018).

[42] Federico Maggi, Marco Balduzzi, Ryan Flores, Lion Gu, and Vincenzo Ciancaglini. Investigating Web Defacement Campaigns at Large. In *Proc. of the ACM Asia Conference on Computer and Communications Security (ASIACCS)* (2018).

[43] Aleecia M. McDonald and Lorrie Faith Cranor. Americans' Attitudes About Internet Behavioral Advertising Practices. In *Proc. of the ACM Workshop on Privacy in the Electronic Society (WPES)* (2010).

[44] Andrey Meshkov. Crypto-Streaming Strikes Back. https://blog.adguard.com/en/crypto-streaming-strikes-back/ (December 2017).

[45] Troy Mursch. Cryptojacking malware Coinhive found on 30,000+ websites. https://badpackets.net/cryptojacking-malware-coinhive-found-on-30000-websites/ (November 2017).

[46] Troy Mursch. How to find cryptojacking malware. https://badpackets.net/how-to-find-cryptojacking-malware/ (February 2018).

[47] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://www.bitcoin.org/bitcoin.pdf (2009).

[48] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You Are What You Include: Large-scale Evaluation of Remote Javascript Inclusions. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2012).

[49] Lindsey O'Donnell. Cryptojacking Attack Found on Los Angeles Times Website. https://threatpost.com/cryptojacking-attack-found-on-los-angeles-times-website/130041/ (February 2018).

[50] Lindsey O'Donnell. Cryptojacking Campaign Exploits Drupal Bug, Over 400 Websites Attacked. https://threatpost.com/cryptojacking-campaign-exploits-drupal-bug-over-400-websites-attacked/131733/ (May 2018).

[51] Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P. Markatos. Truth in Web Mining: Measuring the Profitability and Cost of Cryptominers as a Web Monetization Model. arXiv:1806.01994v1 [cs.CR] (June 2018).

[52] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. The Cost of Digital Advertisement: Comparing User and Advertiser Views. In *Proc. of the World Wide Web Conference (WWW)* (2018).

[53] Giancarlo Pellegrino, Christian Rossow, Fabrice J. Ryba, Thomas C. Schmidt, and Matthias Wählisch. Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics. In *Proc. of the USENIX Workshop on Offensive Technologies (WOOT)* (2015).

[54] Pirate Bay. Miner. https://thepiratebay.org/blog/242 (September 2017).

[55] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *Proc. of the USENIX Security Symposium* (2008).

[56] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The Ghost in the Browser Analysis of Web-based Malware. In *Proc. of the Workshop on Hot Topics in Understanding Botnets (HotBots)* (2007).

[57] Jan Rüth, Torsten Zimmermann, Konrad Wolsing, and Oliver Hohlfeld. Digging into Browser-based Crypto Mining. In *Proc. of the ACM Internet Measurement Conference (IMC)* (2018). (Preprint: https://arxiv.org/abs/1808.00811v1).

[58] Salon. FAQ: What happens when I choose to "Suppress Ads" on Salon? https://www.salon.com/about/faq-what-happens-when-i-choose-to-suppress-ads-on-salon (2018).

[59] Jérôme Segura. Malicious cryptomining and the blacklist conundrum. https://blog.malwarebytes.com/threat-analysis/2018/03/malicious-cryptomining-and-the-blacklist-conundrum/ (March 2018).

[60] Jérôme Segura. The state of malicious cryptomining. https://blog.malwarebytes.com/cybercrime/2018/02/state-malicious-cryptomining/ (March 2018).

[61] Seigen, Max Jameson, Tuomo Nieminen, Neocortex, and Antonio M. Juarez. CryptoNight Hash Function. https://cryptonote.org/cns/cns008.txt (March 2013).

[62] Denis Sinegubko. Hacked Websites Mine Cryptocurrencies. https://blog.sucuri.net/2017/09/hacked-websites-mine-crypocurrencies.html (September 2017).

[63] Slushpool. Stratum Mining Protocol. https://slushpool.com/help/manual/stratum-protocol (2016).

[64] Rashid Tahir, Muhammad Huzaifa, Anupam Das, Mohammad Ahmad, Carl Gunter, Fareed Zaffar, Matthew Caesar, and Nikita Borisov. Mining on Someone Else's Dime: Mitigating Covert Mining Operations in Clouds and Enterprises. In *Proc. of the International Symposium on Recent Advances in Intrusion Detection (RAID)* (2017).

[65] Iain Thomson. Pulitzer-winning website Politifact hacked to mine crypto-coins in browsers. https://www.theregister.co.uk/2017/10/13/politifact_mining_cryptocurrency/ (October 2017).

[66] Mircea Trofin. Chromium Code Reviews Issue 2656103003: [wasm] flag for asm-wasm investigations. https://codereview.chromium.org/2656103003/ (January 2017).

[67] Alejandro Viquez. Opera introduces bitcoin mining protection in all mobile browsers – here's how we did it. https://blogs.opera.com/mobile/2018/01/opera-introduces-bitcoin-mining-protection-mobile-browsers/ (January 2018).

[68] Luke Wagner. Turbocharging the Web. *IEEE Spectrum* (December 2017). (Online version: https://spectrum.ieee.org/computing/software/webassembly-will-finally-let-you-run-highperformance-applications-in-your-browser/).

[69] Wenhao Wang, Benjamin Ferrell, Xiaoyang Xu, Kevin W. Hamlen, and Shuang Hao. SEISMIC: SEcure In-lined Script Monitors for Interrupting Cryptojacks. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)* (2018).

[70] Web Hypertext Application Technology Working Group. HTML Living Standard: Web workers. https://html.spec.whatwg.org/multipage/workers.html (2018).

[71] Chris Williams. UK ICO, USCourts.gov... Thousands of websites hijacked by hidden crypto-mining code after popular plugin pwned. http://www.theregister.co.uk/2018/02/11/browsealoud_compromised_coinhive/ (February 2018).

[72] Dongpeng Xu, Jiang Ming, and Dinghao Wu. Cryptographic Function Detection in Obfuscated Binaries via Bit-Precise Symbolic Loop Mapping. In *Proc. of the IEEE Symposium on Security and Privacy (S&P)* (2017).

[73] Yandex. Yandex Browser Strengthens Cryptocurrency Mining Protection. https://yandex.com/company/blog/yandex-browser-strengthens-cryptocurrency-mining-protection/ (March 2018).

[74] Zhang Zaifeng. Who is Stealing My Power III: An Adnetwork Company Case Study. https://blog.netlab.360.com/who-is-stealing-my-power-iii-an-adnetwork-company-case-study-en/ (February 2018).

[75] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The Dark Alleys of Madison Avenue: Understanding Malicious Advertisements. In *Proc. of the ACM Internet Measurement Conference (IMC)* (2014).

[76] Tianwei Zhang, Yinqian Zhang, and Ruby B. Lee. CloudRadar: A Real-Time Side-Channel Attack Detection System in Clouds. In *Proc. of the International Symposium on Recent Advances in Intrusion Detection (RAID)* (2016).

[77] Zeljka Zorz. How a URL shortener allows malicious actors to hijack visitors' CPU power. https://www.helpnetsecurity.com/2018/05/23/url-shortener-cryptojacking/ (May 2018).