# The Other Side of the Coin: A Framework for Detecting and Analyzing Web-based Cryptocurrency Mining Campaigns

Julian Rauchberger[1], Sebastian Schrittwieser[2], Tobias Dam[1], Robert Luh[2], Damjan Buhov[2], Gerhard Pötzelsberger[1] and Hyoungshick Kim[3]
[1]Institute for IT Security Research, St. Pölten UAS, Austria
[2]Josef Ressel Center TARGET, St. Pölten UAS, Austria
[3]Sungkyunkwan University, Republic of Korea
[1, 2]{first.last}@fhstp.ac.at
[3]hyoung@skku.edu

**ACM Reference Format:**
Julian Rauchberger[1], Sebastian Schrittwieser[2], Tobias Dam[1], Robert Luh[2], Damjan Buhov[2], Gerhard Pötzelsberger[1] and Hyoungshick Kim[3] [1]Institute for IT Security Research, St. Pölten UAS, Austria [2]Josef Ressel Center TARGET, St. Pölten UAS, Austria [3]Sungkyunkwan University, Republic of Korea [1, 2]{first.last}@fhstp.ac.at [3]hyoung@skku.edu . 2018. The Other Side of the Coin: A Framework for Detecting and Analyzing Web-based Cryptocurrency Mining Campaigns. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3230833.3230869

## ABSTRACT

Mining for cryptocurrencies is usually performed on high-performance single purpose hardware or GPUs. However, mining can be easily parallelized and distributed over many less powerful systems. Cryptojacking is a new threat on the Internet and describes code included in websites that uses a visitor's CPU to mine for cryptocurrencies without the their consent. This paper introduces MiningHunter, a novel web crawling framework which is able to detect mining scripts even if they obfuscate their malicious activities. We scanned the Alexa Top 1 million websites for cryptojacking, collected more than 13,400,000 unique JavaScript files with a total size of 246 GB and found that 3,178 websites perform cryptocurrency mining without their visitors' consent. Furthermore, MiningHunter can be used to provide an in-depth analysis of cryptojacking campaigns. To show the feasibility of the proposed framework, three of such campaigns are examined in detail. Our results provide the most comprehensive analysis to date of the spread of cryptojacking on the Internet.

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3230869

## 1 INTRODUCTION

A fundamental building block of cryptocurrencies is *mining* [3], a resource-intensive process that typically requires massive processing power for solving a computational puzzle (e.g., computing hashes on inputs from a large search space in a brute force manner) to maintain a blockchain by adding the next valid block. For mining cryptocurrencies such as Bitcoin [18] or Ethereum [24], high-performance GPUs or even application-specific integrated circuits (ASICs) are required to achieve profitable mining outputs when considering the electricity costs resulting from the power consumption of the mining process. In some cases, however, mining can be easily parallelized and distributed over many less powerful systems. Miners often connect to so-called mining pools [14] to combine their computational power and make mining outputs more predictable. Recently, the dramatic increase of cryptocurrency price attracts malware authors to use victims' devices for covert cryptocurrency mining. While not as resource effective as native mining malware, cryptocurrencies can also be mined in the web browser through JavaScript. Browser-based mining dates back to 2011 when a website called BitcoinPlus.com was launched [12]. This benign service mined for Bitcoins, and even though the computational mining difficulty was relatively low at that time, the reward was tiny compared to the amount of mining power and thus electricity required. Therefore, browser-based mining services quickly disappeared due to their the lack of economic incentives.

However, with rising cryptocurrency rates in recent months and the launch of a new browser-based mining service called *Coinhive* in September 2017, browser-based cryptocurrency mining activities increased significantly. Instead of Bitcoin, which is profitable on dedicated mining hardware only, the current browser-based miners almost exclusively focus on another popular cryptocurrency called Monero. This privacy-focused cryptocurrency [22] was introduced in 2014 and is very well suited for CPU mining. Coinhive is marketed as a revenue alternative to traditional advertising. Their concept is based on providing an alternative way to pay for web-content by using the viewer's CPU resources for performing mining activities after loading the website. In contrast, attacks where visitors of a site are turned into cryptocurrency miners without their knowledge or consent are called *cryptojacking* [7].
One of the first high-profile sites which started using Coinhive mining without their visitors' consent was the well-known torrent website called *The Pirate Bay.* Since then, many other websites have started to use Coinhive mining scripts for their profit.

Other incidents of cryptojacking included embedded video players[1], browser extensions[2], and WordPress plugins[3]. In December 2017, Coinhive was the most prevalent threat of Check Point's Global Threat Index [4]. According to this report, crypto-miners managed to impact 55% of organizations globally, with two variants, Coinhive and *Crypto-Loot*, in the top three list of malware. Ad blockers such as uBlock Origin [9] already implemented basic cryptojacking mitigations and also dedicated browser extensions such as NoCoin [11] were introduced to block unwanted mining activities. Usually, these extensions use manually crafted blacklists to block known cryptojacking scripts. However, in response to these blockers, crypto miners have tried to obfuscate their mining scripts to hide them from cryptocurrency mining detection techniques.

To the best of our knowledge, this work is the first to academically address cryptojacking. We introduce MiningHunter, a novel web crawling framework which allows us to identify and analyze entire cryptojacking campaigns spread over multiple websites. In the Alexa Top 1 million list of websites [1], we were able to identify 3,000 websites with active mining activities and 5 big campaigns with more than 50 websites infected through each. Our results draw the most complete picture of the cryptojacking economy on the web up to date and we also hope our work will stimulate future academic research in the novel field of browser-based cryptocurrency mining. In particular, the main contributions of this paper are:

- We introduce MiningHunter, a novel web crawling framework based on Chrome DevTools Protocol, that is able to reliably identify mining activities on websites though WebSocket traffic and other indicators.
- We present the first comprehensive scientific study on browser-based cryptocurrency mining campaigns for the Alexa Top 1 million websites.
- To demonstrate the power of our framework, we discuss the finding regarding three mining campaigns in detail and describe the big picture of today's state of cryptojacking on the web.

The remainder of this paper is structured as follows: In Section 2, the principles of browser-based mining is described. Section 3 introduces our web crawling framework MiningHunter. In Section 4, the results of our research are presented, whereas Section 5 evaluates the outcome and presents an in-depth analysis of three mining campaigns. Section 6 discusses related work and Section 7 concludes the paper.

## 2 BACKGROUND

Coinhive, the most porpularly used malware for cryptojacking, enables website owners to easily integrate cryptocurrency mining into their website. It is packaged into a *JavaScript API* which contains the preferred configuration for the operation. When the website is accessed by a user, the mining process is started in the background. In contrast, their later introduced service called authedmine.com requires explicit permission from the users in order to start mining.

First the Coinhive JavaScript file is executed by the browser. The script can either be fetched from

[1]https://blog.adguard.com/en/crypto-streaming-strikes-back/
[2]https://www.bleepingcomputer.com/news/security/chrome-extension-with-100-000-users-caught-pushing-cryptocurrency-miner/
[3]https://www.wordfence.com/blog/2017/11/wordpress-plugin-banned-crypto-mining/

coinhive.com or a self-hosted location. Next the mining threads are started and a persistent WebSocket connection using the WebSocket API is created. After authenticating to the Coinhive server, the actual mining tasks are performed. New proof-of-work jobs are received, executed locally and the results are reported back to the server via the WebSocket connection. Due to the emergence of blocking add-ons such as NoCoin which apply domain based detection of mining, proxy services (e.g. Stratum mining proxy[4]) became popular for hiding the connection to Coinhive. Proxies not only render ad blockers completely ineffective, but allow website owners to circumvent Coinhive completely and use any mining pool they want. This also eliminates the obligatory fees Coinhive charges its users.

Today Coinhive is almost exclusively used for mining *Monero*, as Monero is very well suited for CPU mining. Monero is based on the CryptoNote protocol [23] and uses the CryptoNight proof-of-work algorithm [21]. CryptoNight is a memory bound function and therefore cannot be as easily pipelined as other proof-of-work algorithms such as Bitcoin's SHA256 based algorithm, which is a CPU-bound function. Monero's proof-of-work algorithm has been designed based on the assumption that the efficiency gap between CPU and GPU/ASIC miners is closed. In the first week after Coinhive was launched the service collectively mined 13.5M hashes per second at peak times, which equaled 5% of the global hash rate of the Monero blockchain at that time [5]. For an estimated average hash rate of 5M hashes per second, the estimated profit of mining Monero was about $8,200 per day, of which Coinhive receives a 30% share of approximately $2,400 per day [6].

## 3 SYSTEM DESIGN

The main goal of MiningHunter was to collect all data required for sophisticated post-scan analysis. This includes metadata about all requested resources as well as the full contents of all executed scripts. We store information about every request that is sent during a scan, including the *initiator* which indicates the exact line of code that caused a given resource to be requested. This allows us to create a chain of events that led to the loading of a specific file and gives us the ability to determine exactly why a mining script had been executed. Since this information (including the exact line number in the source code for triggering each request) was collected directly from the browser, the constructed resource chains are more complete than in those obtained by other frameworks (e.g., [13, 15]) which mainly use the HTTP Referer header or manual parsing of JavaScript code. In order to reliably detect cryptocurrency miners in our large dataset without relying on manually created blacklists, we developed a dynamic detection approach. The Coinhive WebSocket traffic seemed to be an ideal target as it consists of JSON objects with a set of name-value pairs with all distinct names, making it easy to fingerprint cryptocurrency miners.

Even though Coinhive seemed to hold a dominant market position taking up the most network traffic, we analyzed a number of smaller competitors to see if the same technique could be employed to detect their miners. We found that a high number of the sites tested (e.g., coin-have.com, crypto-loot.com, minero.pw and moneroMiner.rocks) were using the exact same protocol, indicating that they can can effectively be detected without any additional changes. Other sites (e.g., webmine.cz) were using a similar JSON-based protocol which we could include with minimal effort.

Our detection method using WebSocket traffic consists of multiple steps. First, we check if the individual frames of the recorded traffic are valid JSON objects that can be parsed and further analyzed. For frames where this is true, we compare the layout of the parsed JSON data with multiple, manually generated fingerprints that match the traffic patterns generated by Coinhive as well as several similar sites. If the generated fingerprint matches with a fingerprint in the database, we store that information in a logfile and increase a counter that indicates how many frames of this particular website matched a fingerprint. MiningHunter also provides a highly flexible way to search through the collected data for additional manual analysis. This allows us to estimate the scope of cryptojacking campaigns that are more sophisticated than a straightforward integration of Coinhive. The data can be searched in several variants, such as:

- regular expressions to search through the URLs of all requests
- grep through all JavaScript code to find files containing specific strings
- search for responses with a specific hash value
- search the WebSocket traffic, e.g. for Coinhive traffic fingerprint

Combining these techniques, we have a sophisticated framework to track cryptocurrency mining campaigns across the Internet. Indicators such as domain names or hashes of scripts can be used to find additional websites

[4]https://github.com/x25/coinhive-stratum-mining-proxy

affiliated with the same miners. This allows us to easily determine the scope of a campaign after manually analyzing a single or very few websites.

## 3.1 MiningHunter

From a high-level perspective, MiningHunter consists of two major parts: an automated browser that scans websites and reports the results back to a central server as well as an orchestration framework that is used to run the infrastructure and to manage the scan jobs and results in the backend. MiningHunter builds upon a headless Chromium which is automated via the Chrome Developer Tools (DevTools) Protocol[5]. This tool was carefully chosen because its interface offers a deeper low-level integration than other browser automation frameworks (e.g., PhantomJS [8]). The DevTools Protocol allows easy access to data such as parsed JavaScript or raw WebSocket traffic. Since Chromium is a standard desktop browser, potentially existing fingerprinting scripts do not have to be tricked into believing that the requests are coming from a real browser. As some mining scripts might target Windows machines only, our Chromium browser runs a custom plugin, which sets the UserAgent and the `navigator.platform` property to the values of a Windows 7 machine. For the implementation of the browser automation, the Chrome Debugging Protocol interface[6] for Node.js is used.

Performing a scan of multiple websites involves several steps as described below. First, a new scan job is received from the backend server. Each job contains a batch of one or more website URLs. The browser then opens a configurable amount of tabs and scans their content simultaneously. Browser tabs are not reused for scanning the next website of the job, so each website is scanned in its own isolated environment. After a specific amount of websites is scanned, the browser is restarted to delete browsing history and cookies. In practice, unrecoverable errors sometimes happen during the scan of a batch of URLs (e.g. when the Chromium browser crashes). In this case, the full job is marked as failed and has to be restarted. All parameters can be configured dynamically while creating the job at the backend server. For the scan of the Alexa Top 1 million, jobs were submitted with a batch size of 10 URLs. The browser was running three concurrent tabs and it was restarted again whenever a batch of 10 URLs was finished. For each scanned website, the following data is collected: Metadata about each request, all executed JavaScript code, and raw WebSocket traffic. The collected metadata is listed in Table 1.

[5]https://chromedevtools.github.io/devtools-protocol/
[6]https://github.com/cyrus-and/chrome-remote-interface

To collect all the executed JavaScript code, the `Debugger.scriptParsed` event of the DevTools Protocol interface is used. It is triggered for all scripts that are parsed by the browser. This also includes `<script>` tags that are embedded into the HTML code of a website. The collected WebSocket traffic includes *connection established* events, all incoming and outgoing traffic, and *connection closed* events. The data collection for a website is stopped when the `Network.loadingFinished` event triggers and a configurable amount of time has elapsed after that event (e.g., one second in our implementation). If the `Network.loadingFinished` event does not trigger within a configurable timespan, data collection is also stopped. This timeout was set to 30 seconds in our scan.

The collected requests, the WebSocket traffic, and the metadata of the collected JavaScript (but not the code itself) are sent in JSON format to the backend server. The server then reads the SHA256 hashes of the JavaScript metadata to determine which code is already stored in the backend server and which is not. Subsequently, the scanner is told by the backend which of the collected JavaScript is new and has to be transmitted to the backend. Deduplication of JavaScript is beneficial for multiple reasons. Rescanning sites becomes faster as most of the JavaScript code will be the same on a repeated scan and does not have to be sent to the backend again. In addition, storage requirements are massively reduced. In our scan, the first 1,000 websites resulted in approximately 1 GB unique JavaScript code, whereas the first 10,000 websites lead to 7 GB, and the first 100,000 websites to 42 GB. The full scan of one million websites resulted in 215 GB unique JavaScript code only. The backend server of MiningHunter is a web application written in Node.js. It uses Kue[7] for job management, which offers an interface to manage jobs and supports automatic retries of failed jobs.

One of the main goals of MiningHunter was scalability. For deployment, Docker is used in swarm mode. The master server runs the backend inside a Docker container, deployed as a Docker stack. The master server also acts as swarm manager. The scanning browser is packed into a Docker container as well and is registered as swarm node. The scanners run on EC2 Spot instances in the Amazon cloud and Docker is able to automatically deploy new scanner containers on them. The amount of scanner containers running on each machine can easily

[7]https://github.com/Automattic/kue

| **Metadata** | **Comment** |
|---|---|
| Request ID | Unique identifier provided by the Chrome DevTools Protocol |
| URL | Full URL of the request including GET parameters |
| Original URL | Only if the request was made due to a redirect |
| Initiator | Either *parser* if HTML code, *script* if JavaScript code, or *other* |
| IP Address of the remote host | Stored in case the domain is no longer resolvable at the time of analysis |
| Status code of response | If available |
| MIME type of responses | If available |
| SHA256 hash | Of the HTTP request's response body |

Table 1: Metadata stored for each request in MiningHunter.

be scaled up and down in Docker, and if a spot instance should crash, the running scanners are automatically distributed to the remaining servers.

## 4 EXPERIMENTS AND RESULTS

For the evaluation of MiningHunter, we scanned the Alexa Top 1 million websites using four *m4.large* spot instances hosted in the Amazon AWS cloud to run the Chromium based scanning containers. The data was then sent back to our backend server hosted locally and stored in flat files. The scan was carried out at the beginning of December 2017 and took approximately 5 days. Using the WebSocket heuristic approach described in Section 3, we detected 3,178 websites that actively ran a script for cryptocurrency mining during our scan. Our scanner generated no user input or other interaction with the sites it scanned, indicating that all of these sites started scanning without requiring explicit consent from users. In fact, we found that the distribution of malware in the wild is extremely biased. Roughly 85.6% of the detected scripts were from Coinhive, 3.8% from CoinHave, 1.9% from Crypto-Loot, and 8.7% from others. We were also able to extract the API keys used to authenticate with the backend server from the recorded WebSocket traffic. In total, we found 1,210 unique keys across 3,178 websites. The most frequently used key was found on 1,116 different websites while 961 keys were detected on a single website only. This shows that the majority of users deploy the scanning script on a single webpage only or opt to use unique API keys for each page. At the same time, there seem to be some bigger operations that deploy their scripts across a large number of websites. More detailed insights on these campaigns gathered from our collected data can be found in Section 5.2.

## 5 EVALUATION

To evaluate the detection rate of the WebSocket fingerprinting technique we developed, we compared MiningHunter's detection capabilities to two popular browser extensions that focus on blocking mining scripts. Table 2 gives on overview on the comparison to the extensions NoCoin [11] and MinerBlock [2].

### 5.1 Comparison to blocklists of browser extensions

During the initial manual analysis with the Alexa Top 10,000 websites only, we found that some miners were already actively attempting to bypass commonly used ad blocking techniques. Very often we would see the mining JavaScript loaded from the website itself rather than Coinhive. Many times, the script had also been renamed from the default coinhive.min.js or directly embedded into the page. Additionally, the WebSockets often connected to IP addresses directly or to another third-party domain which had seemingly only been set up to forward Coinhive WebSocket traffic to another mining pool. As most commonly used ad blocking browser extensions are making extensive use of domain or path-based blacklisting, it seemed very likely that these simple re-hosting and traffic proxying tricks were intended to bypass these blacklists and avoid getting blocked. In many cases, this seemed to work as general-purpose ad blocking extensions failed to detect the miners. In order to evaluate the capabilities of our heuristic WebSocket traffic fingerprinting approach, we compare the detection results of our method to two browser extensions specializing in blocking cryptocurrency miners, NoCoin and MinerBlock.

#### 5.1.1 *Heuristic.*

To detect cryptocurrency miners based on WebSocket traffic, we fully recorded all incoming and outgoing WebSocket traffic during our scan of the Alexa Top 1 million. To detect sites which mine cryptocurrency, we developed a fingerprinting-based approach that can reliably detect Coinhive WebSocket traffic based on the layout of the JSON messages. For the results in this paper, we opted to use very strict heuristics to avoid false positives. At

| | **Active mining** | **Unique Findings** | **No WebSocket Traffic** | **WebSocket opened, but no mining** |
|---|---|---|---|---|
| MiningHunter | 3,178 | 261 | 0 | 0 |
| NoCoin | 2,883 | 48 | 295 | 133 |
| MinerBlock | 2,885 | | 217 | 133 |

Table 2: Comparison of websites detected by our dynamic detection approach and the browser extensions NoCoin and MinerBlock. The *Unique Findings* column shows the number of websites our dynamic approach found compared to the two browser extensions combined and vice versa. The *No WebSocket Traffic* column shows the number of websites contained in the blacklists of the browser extensions which did not show any WebSocket traffic in our scans. These websites could either be using mining scripts that do not rely on WebSockets or simply be false positives. The *WebSocket opened, but no mining* column shows the number of websites that opened a WebSocket connection, but did not start mining in our scans.

the same time, it might be possible that some mining scripts which added small modifications to the protocol were not detected. With our dynamic approach, we were able to identify 3,178 sites which actively tried to mine cryptocurrencies.

### 5.1.2 *NoCoin.*

NoCoin [11] is a popular browser extension which blocks cryptocurrency miners based on a blacklist which is maintained and regularly updated by the developers themselves. The project is open source and can be found on GitHub. To compare the performance of NoCoin to our dynamic approach, we downloaded the blocklist [8], converted it to equivalent regular expressions and ran them against the request URLs and WebSocket URLs in all of our scans. Whenever at least one regular expression pattern was matched, we counted the website as cryptocurrency miner. NoCoin detected the presence of mining scripts on 3,311 websites. We compared those websites to the ones our heuristic detected and found that 327 were only found by our approach while 460 websites were only detected by NoCoin. The rest was found by both.

To evaluate the shortcomings of our heuristic, we further analyzed the 460 sites which we did not detect and found that among those websites, 295 had no WebSocket traffic at all. It could be possible that these sites loaded a known mining script but did not start because users were explicitly asked to agree a consent form. Our dynamic approach can only detect sites that automatically start mining without explicit user consent. Alternatively, the miners could function without the use of WebSockets at all which we cannot detect as we only record WebSocket traffic. This was mostly due to our limited resources for transmitting and storing scan results. It would also be possible to retain a copy of every request and response body which would allow for the detection of these miners. Additionally, 133 of the sites opened a WebSocket connection to either coinhive.com or crypto-loot.com but did not send any traffic. Most of the sites attempted to open multiple connections in rapid succession, indicating that there might have been some connection issues at the time of the scan. Alternatively, the sites might have loaded so slowly that the scan stopped before the mining process started. As our heuristic correctly detected Crypto-Loot as well as Coinhive traffic on many other sites, it is safe to assume that it would have correctly identified these sites as well if they had sent any traffic. This leaves only 32 sites sending WebSocket traffic which we did not detect. These results imply that the number of sites potentially missed by our WebSocket heuristic is very small.

### 5.1.3 *MinerBlock.*

MinerBlock [2] is an extension similar to NoCoin that employs its own blacklist. It also has some simple heuristics to detect cryptocurrency miners at runtime but we here focus on the blacklist approach for comparison. As with NoCoin, we downloaded the publicly available blacklist of MinerBlock from GitHub[9], converted it to equivalent regular expressions and used them to filter all requests in our dataset. This resulted in 3,235 websites where mining scripts were detected. Comparing those to the results of our dynamic approach, 323 sites were only found by our heuristic and 380 were only found by the MinerBlock blacklist. Again, we investigated the sites our approach did not detect. Among the 380 sites only detected by the blacklist, 217 sites did not have any WebSocket traffic at all (for possible reasons please refer to Section 5.1.2). MinerBlock also detected the

[8]https://github.com/keraf/NoCoin/blob/master/src/blacklist.txt

[9]https://github.com/xd4rker/MinerBlock/blob/master/assets/filters.txt

same 133 sites that opened the WebSocket connection to coinhive.com or miner-block.com but did not send any further traffic. Again, such behaviors were already discussed in Section 5.1.2. Finally, 30 undetected sites did have unrecognized WebSocket traffic, again showing similar results to our analysis of the NoCoin blacklist.

According to our experiment results, our dynamic approach performed very similar to both the NoCoin and MinerBlock blacklists. Taking into account only the websites which make use of WebSockets, we were able to detect around 99% of the sites found by the manual blacklists. Right now, obfuscation of WebSocket traffic does not seem to be very widespread while attempts to bypass blacklists seem to be more common. This is directly reflected by the websites our dynamic approach found but the blacklists missed. Results from the NoCoin and MinerBlock blacklists show that the vast majority of crypto miners found in the wild use a WebSocket-based protocol similar or equal to Coinhive. Cryptocurrency miners that do not make use of WebSockets seem to be a small minority as less than 10% of the sites detected by blacklists had no WebSocket traffic. These also contain the websites that did not start mining for other reasons such as user consent, so the real number of non-WebSocket based miners is likely even lower.

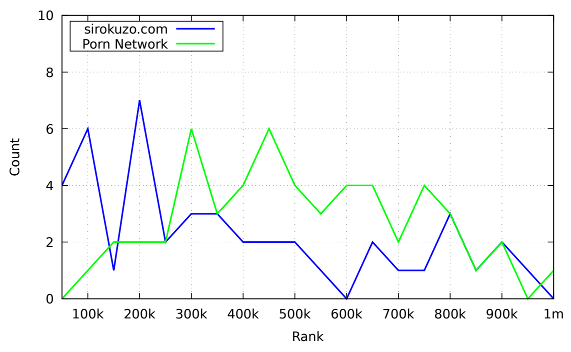

Figure 1: Distribution of mining campaigns 1 and 3 over the Alexa Top 1 million sites

For the purpose of our research, a clear advantage is that with our dynamic approach we can be sure that the sites in question did actually perform unauthorized mining activities. With blacklists, we would only be able to determine which sites load mining scripts but they might still ask for user consent or not start mining for other reasons. Furthermore, the fingerprinting is not influenced by domain names or proxies which means the results are not skewed towards more popular sites which are more likely to be included in manually created blacklists. This gives us a more realistic view of the distribution of miners over the Alexa Top 1 million.

## 5.2 Mining campaigns

We evaluated our approach by manual analysis of three of the identified mining campaigns. Figure 1 illustrates the distribution of websites contained in the Campains 1 and 3 over the Alexa Top 1 million ranks condensed into groups of 50,000. Campaign 2 is depicted separately in Figure 2.

### 5.2.1 *Campaign 1: sirokuzo.com.*

We initially detected this campaign when we were manually analyzing JavaScript files of a preliminary Alexa Top 10,000 scan. After searching all of our stored files for the string cryptonight - the name of an algorithm used in mining different cryptocurrencies, including Monero - we detected multiple oddly named files hosted on the domain sirokuzo.com (1011165a059.1.n.2.1.js, 7751911085a11c28.2.n.2.1.js, 4665c028b22f47978cb0 b5d4d39e66cb.wasm.js).

Upon further inspection of the domains which loaded the suspicious files, we concluded that these files do indeed mine cryptocurrencies without user consent as CPU usage spiked and web workers were created when visiting the infected websites. We also observed WebSocket traffic to another third-party domain, salamalyum.com. Unlike all other mining traffic we had come across before, this one consisted entirely of Base64 encoded strings which would only produce gibberish when decoded, a clear sign of encryption. The code of the miner itself seemed to be highly obfuscated with an algorithm more advanced than what we typically see. Additionally, file contents as well as names of the mining scripts seemed to be unique for each domain that loaded them, indicating that the authors were looking to bypass simple URL or content-based blocking techniques.

After some manual analysis, we found that the code had most likely been obfuscated with an open source project called javascript-obfuscator[10]. The algorithm, however, is not perfect and leaves some underlying strings unobfuscated. We dynamically debugged the code to dump function names and other strings at runtime. Searching GitHub for these unique strings showed that large parts of the obfuscated code were actually based on the crypto-js project which is also open source[11]. With this information we were able to pinpoint the actual main logic which is in itself quite small as most of the obfuscated code is part of the crypto-js library. It encrypts all traffic sent over the WebSocket with AES using the default settings of crypto-js. With the algorithm known, we added code

[10]https://github.com/javascript-obfuscator/javascript-obfuscator
[11]https://code.google.com/archive/p/crypto-js/

to MiningHunter that is able to decrypt the traffic if the key is provided. To find the encryption key, we used a JavaScript debugger to step through the code and print the key after it is loaded into memory. With this technique, we were also able to ascertain the Monero wallet where the mined funds were sent to. Based on the samples we analyzed, the key seems to be static and the same across all websites. We then applied the analysis script to the full Alexa 1 million scan and found 46 domains in total which had WebSocket traffic that could be decrypted with the sirokuzo algorithm and key.

We then used MiningHunter to find all other JavaScript files which were hosted on the same domain as the obfuscated mining scripts. In one of those we found partially unobfuscated code that still retained human readable references to the domain gridcash.net. By registering an account for their service we were able to confirm that the code they supply to their customers was indeed the same we originally found on sirokuzo.com even though this domain is no longer used and has long since been switched to a new one. This example shows us that at least some providers of cryptojacking services are spending considerable resources to avoid being blocked by common ad blocking techniques. Using unique filenames, obfuscating script contents on a per-domain basis, and regularly switching the domain which hosts scripts are all techniques commonly used by malicious actors to bypass antivirus protection. This indicates that simple block lists of URLs or file content hashes might no longer be enough to stop cryptojacking. Additionally, AES encryption was used to obfuscate the mining protocol. To the best of our knowledge, there is no publicly available software that blocks miners based on WebSocket traffic, so this seems to indicate that the authors correctly identified this as a weak spot that could be used to detect their miner in the future.

MiningHunter has many features which were helpful in determining the scope of this campaign. After manually analyzing a single case, we were easily able to find further affected domains by simply searching for other websites that request resources from sirokuzo.com. This gave us the necessary sample size to ascertain that the same encryption key is most likely used on all these websites. Since MiningHunter stores the full WebSocket traffic, we could then develop a decryption algorithm and find affected websites that load the JavaScript from domains other than sirokuzo.com. By using MiningHunter to search all requests to these sites, we were able to find other JavaScript files also hosted there. This gave us the information necessary to trace the campaign back to gridcash.net.

#### 5.2.2 *Campaign 2: adsco.re.*

Since we store the full WebSocket traffic of all sites we scanned, we wrote a simple heuristic that is able to detect JSON-based communication which looks similar to Coinhive traffic. This traffic also contains the API key which the miner uses to identify itself with the server. Coinhive uses this to track for which of their affiliates each user is mining. Analyzing the appearance of these keys, we can group together single instances of mining on seemingly unassociated websites to the same actor. Counting the appearance of each API key, we found a single one which appeared much more often than every other key. This key appeared on over 1,000 different websites, with the second most found key only appearing on 90.

On the first glance, these websites had nothing in common and clearly did not belong to the same company. This made it highly unlikely that all cryptojacking scripts had been manually embedded and configured with the same API key. We then decided to use our tracing script to determine how the mining script was actually loaded on these pages and found that in every instance, they were loaded via two third-party domains. The affected sites initially load pop.js from c1.popads.net which loads another script from serve.popads.net. This script then loads content from c.adsco.re which hosts another script that finally includes the mining script from coinhive.com.

From this chain, it becomes clear that the adsco.re domain is part of a malvertising campaign [26] that distributes malicious JavaScript through ads on popads.net. We used MiningHunter to analyze if the same adsco.re script is also hosted on other domains but found no results. Similarly, we looked for additional scripts loaded from adsco.re but also found nothing. This indicates that the campaign was limited to this single domain at the time of the scan. Figure 2 shows the distribution of websites affected by the mining campaign *adsco.re* over the Alexa Top 1 million ranks condensed into groups of 10,000.

The adsco.re campaign shows that there are actors willing to put cryptocurrency miners into advertisements. Even if a user never visits websites that mine by themselves, they might still be exposed to cryptojacking via malicious ads. It is still unclear how profitable these operations are since the advertisements should incur significant costs themselves. Even if it might not be lucrative in itself, we still expect to see more of these cases in the future as advertisers could include mining

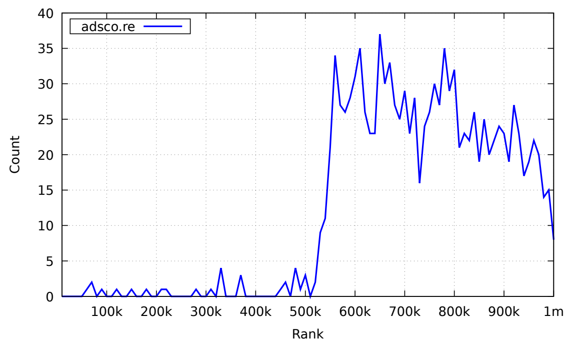

Figure 2: Distribution of mining campaign 2 over the Alexa Top 1 million sites

scripts with legitimate ads to earn some additional income. Since they would have likely bought the advertisements anyway, they lose nothing by including a cryptominer and are able to reduce overall costs. It could also be possible that advertisement networks include the mining scripts themselves in every ad to improve revenue.

#### 5.2.3 *Campaign 3: Porn network.*

When analyzing the occurrence of API keys for the adsco.re campaign, we came across an unrelated but very unusual API key. This particular key had been found on over 50 sites, but was more than twice as long than the usual Coinhive keys. It seemed very likely that this key belonged to another cryptojacking provider altogether, so we decided to investigate this case further. A manual analysis of an affected site showed us that their mining setup was based on the standard Coinhive JavaScript but additional code had been added that redirects the WebSocket traffic to another IP address with the URL ending in /proxy. It seems very likely that they are making use of one of the many Coinhive Stratum proxy open source projects found on various sites such as GitHub. They enable the use of the mining JavaScript from Coinhive with any mining pool that supports the Stratum protocol, thereby avoiding the fees of Coinhive at the cost of having to run your own proxy.

When we used MiningHunter to investigate how the mining script was being loaded by the affected pages, we found some similarities between all of them. They all load the script through the same third-party domain and this domain also hosts other, legitimate resources such as images and fonts for many of the sites. Furthermore, we found no signs that the websites had been compromised and also no attempts to hide the presence of the mining script had been made. Finally, all affected websites hosted content of pornographic nature and shared some similarities regarding layout and website design. This lead us to conclude that the mining script had been placed deliberately by the administrator. The results in this case show us that browser-based cryptocurrency mining also seems to be embraced by site owners trying to improve their revenue and not only by cybercriminals. It should however be noted that the pages we checked manually still contained normal advertisements in addition to the cryptojacking code.

## 6 RELATED WORK

At the time of writing no academic publication on cryptojacking existed. However, several reports and web blogs covered the topic. Jereme Segura (a.k.a. malwarebytes) [20] analyzed the introduction of browser-based cryptocurrency mining as a monetizing option for websites and the almost immediate abuse of the technology though cryptojacking, which is referred to as *drive-by cryptocurrency mining*. The technical report primarily focuses on Coinhive. Segura also noted that as soon as crypto mining blockers were available, operators of cryptojacking implemented proxy-based mitigations immediately.

In October 2017, AdGuard conducted a study on the Alexa Top 100,000 websites searching for the implementation of Coinhive and JSEcoin [16]. They found crypto mining scripts on 220 websites with an aggregated audience of 500 million users. In an updated study one month later, AdGuard found that browser mining grew by 31% over the past month on the Alexa Top 100,000 websites [17]. Overall they found over 33,000 websites loading cryptojacking scripts, with a total traffic of 1 billion monthly visits. Almost 95% of the websites they found ran the Coinhive script. Within the Alexa Top 1 million they found only around 1,500 infected websites, compared to our 3,000 findings in December 2017. As AdGuard's methodology is not publicly described, we are not able to compare it to ours from a technical perspective. Abusing computing resources of others for mining purposes without consent is also possible outside the web browser. Plohmann et al. [19], Huang et al. [10], and Wyke [25] analyzed botnets that were used for mining Bitcoins on infected machines.

## 7 CONCLUSION

This paper introduced MiningHunter, a novel web crawling framework for detecting web-based cryptocurrency mining, and presented an in-depth analysis of the current state of cryptojacking on the Internet. Mining cryptocurrencies in the web browser of a website's visitor without their consent is an emerging problem and its detection gets increasingly difficult with the wider use of proxy servers and mining script obfuscation techniques. Our approach of looking for dynamic artifacts of the mining scripts largely mitigates this problem. In early December 2017 we were able to identify 3,178 websites with active mining scripts that do not ask the user for permission before starting the mining process.

MiningHunter also makes it possible to perform in-depth analyses of mining campaigns. We found several large-scale campaigns, one of which encompassed over 1,000 infected sites and even delivered their mining script through malicious advertising. Current blocking extensions for web browsers are able to accurately detect connections to Coinhive, the biggest browser-mining provider to date, and similar sites, but often fail at obfuscated scripts as well as proxy servers. MiningHunter is able to automatically perform large-scale scans and can be used to generate better blacklists than existing solutions.

## ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

## REFERENCES

[1] Alexa Top 1 Million Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.
[2] Belkacim, I. MinerBlock Browser Extension. https://github.com/xd4rker/MinerBlock.
[3] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 104–121.
[4] Check Point Research Team. December's Most Wanted Malware: Crypto-Miners Affect 55% of Businesses Worldwide, 2018. https://blog.checkpoint.com/2018/01/15/decembers-wanted-malware-crypto-miners-affect-55-businesses-worldwide/.
[5] Coinhive. First week status report, 2017. https://coinhive.com/blog/status-report.
[6] Davidi, A. Don't Mine Me Coinhive, 2017. https://www.trustwave.com/Resources/SpiderLabs-Blog/%E2%80%9CDon-t-Mine-Me%E2%80%9D-%E2%80%93-Coinhive/.
[7] European Union Agency for Network and Information Security (ENISA). Cryptojacking – Cryptomining in the browser, 2017. https://www.enisa.europa.eu/publications/info-notes/cryptojacking-cryptomining-in-the-browser.
[8] Hidayat, A. Phantomjs: headless webkit with javascript api. *WSEAS Trans. Commun* (2013), 457–477.
[9] Hill, R. ublock Origin Browser Extension. https://github.com/gorhill/uBlock/.
[10] Huang, D. Y., Dharmdasani, H., Meiklejohn, S., Dave, V., Grier, C., McCoy, D., Savage, S., Weaver, N., Snoeren, A. C., and Levchenko, K. Botcoin: Monetizing stolen cycles. In *NDSS* (2014).
[11] Keramidas, R. NoCoin Browser Extension. https://github.com/keraf/NoCoin.
[12] Lau, H. Browser-based cryptocurrency mining makes unexpected return from the dead, 2017. https://www.symantec.com/blogs/threat-intelligence/browser-mining-cryptocurrency.
[13] Levchenko, K., Pitsillidis, A., Chachra, N., Enright, B., Félegyházi, M., Grier, C., Halvorson, T., Kanich, C., Kreibich, C., Liu, H., et al. Click trajectories: End-to-end analysis of the spam value chain. In *Security and Privacy (SP), 2011 IEEE Symposium on* (2011), IEEE, pp. 431–446.
[14] Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., and Rosenschein, J. S. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), International Foundation for Autonomous Agents and Multiagent Systems, pp. 919–927.
[15] Li, Z., Zhang, K., Xie, Y., Yu, F., and Wang, X. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security* (2012), ACM, pp. 674–686.
[16] Meshkov, A. Cryptocurrency mining affects over 500 million people. And they have no idea it is happening., 2017. https://blog.adguard.com/en/crypto-mining-fever/.
[17] Meshkov, A. Cryptojacking surges in popularity growing by 31% over the past month, 2017. https://blog.adguard.com/en/november_mining_stats/.
[18] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, 2008.
[19] Plohmann, D., and Gerhards-Padilla, E. Case study of the miner botnet. In *Cyber Conflict (CYCON), 2012 4th International Conference on* (2012), IEEE, pp. 1–16.
[20] Segura, J. A look into the global drive-by cryptocurrency mining phenomenon, 2017. https://go.malwarebytes.com/rs/805-USG-300/images/Drive-by_Mining_FINAL.pdf.
[21] Seigen, Jameson, M., Nieminen, T., Neocortex, and Juarez, A. M. CryptoNight Hash Function, 2013. https://cryptonote.org/cns/cns008.txt.
[22] Sun, S.-F., Au, M. H., Liu, J. K., and Yuen, T. H. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *European Symposium on Research in Computer Security* (2017), Springer, pp. 456–474.
[23] Van Saberhagen, N. Cryptonote v 2. 0, 2013. https://cryptonote.org/whitepaper.pdf.
[24] Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper 151* (2014), 1–32.
[25] Wyke, J. The zeroaccess botnet–mining and fraud for massive financial gain. *Sophos Technical Paper* (2012).
[26] Zarras, A., Kapravelos, A., Stringhini, G., Holz, T., Kruegel, C., and Vigna, G. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (New York, NY, USA, 2014), IMC '14, ACM, pp. 373–380.