George: An Intelligent Tool to Detect Cryptojacking

Aman Raj
aman.k.raj@stonybrook.edu
Stony Brook University

Rishabh Jain
rishabh.jain@stonybrook.edu
Stony Brook University

Digant Jain
digant.jain@stonybrook.edu
Stony Brook University

I ABSTRACT

In today's world increasing number of websites are switching from traditional methods of displaying advertisements as a mechanism of generating revenues to mining cryptocurrency leveraging user's machine power. Some websites display warnings and asks for the user's approval before mining currency, but others mine without the consent of the user. We have conducted a survey for the top 250,000 of Alexa top 1 million websites to find how many of them have been using this new methodology to earn revenues for them. After our analysis we found ∼120 (0.05%) websites of the Alexa top 250,000 websites were indulged in cryptojacking.

Till this day there has been a development of various browser plugins and softwares to detect and block cryptocurrency mining sites. Though there have been some studies conducted to give an estimate of how many websites are indulged in cryptojacking, no such algorithm has been devised to predict whether any website is mining cryptocurrency. In our implementation, we have used the fact that browser uses a lot of CPU cycles for the mining of cryptocurrency. We have implemented machine learning algorithm that uses the CPU usage of a system as the input feature for predicting whether a website is mining cryptocurrency or not with a confidence score. We then used our algorithm to predict for 50 random websites whether they had been mining or not and obtained an accuracy of almost 99%.

II INTRODUCTION

Cryptojacking is a form of cyberattack in which a hacker hijacks a target's processing power in order to mine cryptocurrency on the hacker's behalf. It is the unauthorized use of someone else's computer to mine cryptocurrency. Hackers do this by either getting the victim to click on a malicious link in an email that loads crypto mining code on the computer, or by infecting a website or online ad with JavaScript code that auto-executes once loaded in the victim's browser. Widely publicized hacks such as the WannaCry worm, which affected systems on several continents in May 2017, encrypted victims' files and demanded cryptocurrency ransoms - Bitcoin, in the case of WannaCry - in order to decrypt them. Cryptojacking takes a different approach, harnessing victims' machines to "mine": perform the computations necessary to update cryptocurrencies' blockchains, creating new tokens and generating fees in the process. These new tokens and fees are deposited to wallets owned by the attacker, while the costs of mining - electricity and wear and tear to computers are borne by the victim.

There have been various instances of cryptojacking in the recent past. As per Investopedia [4], In February 2018, a Spanish cybersecurity firm, Panda, wrote that a cryptojacking scrypt known as WannaMine had spread to "computers around the world." The malware was being used to mine monero, a cryptocurrency that is notable for its ability to mine using CPUs (as opposed to GPUs or ASICs) while actually having some value in fiat terms. Later the same month, governments in Britain, the U.S. and Canada were affected by a cryptojacking attack that took advantage of a vulnerability in a text-to-speech software embedded in many of these governments' sites. Attackers inserted Coinhive script into the software, allowing them to mine monero using visitors' browsers. Later in February, it was revealed that Tesla Inc. had been the victim of cryptojacking when its Amazon Web Services software container was compromised. Similar attacks on companies have been reported going back to October 2017.

Browser mining is becoming an increasingly common practice. The lines between cryptojacking and legitimate practice are not always clear. Coinhive is often described as malware, but Salon recently partnered with its developers to mine monero using visitors' browsers - with their permission – as a way of monetizing the outlet's content when faced with adblockers. Some experts have cited the potential of browser mining as an alternative to ad-based monetization: in essence, legitimized cryptojacking. Such proposals are extremely controversial, given the potential costs to users in terms of power consumption and damage to hardware.

Currently, there are various ways that exist to minimize the risk of Cryptojacking.

- **Install an ad-blocking or anti-cryptomining extension on web browsers**: Since cryptojacking scripts are often delivered through web ads, installing an ad blocker can be an effective means of stopping them. Some ad blockers like Ad Blocker Plus have some capability to detect crypto mining scripts. Extensions like No Coin and MinerBlock, which are designed to detect and block cryptomining scripts are also helpful.
- **Using endpoint protection that is capable of detecting known crypto miners**: Many of the endpoint

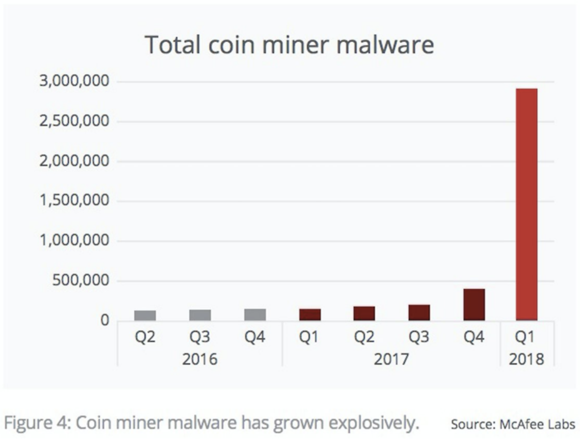


Fig. 1: The above image shows the dramatic increase of the miner malware, specially in Q1 2018. This is the time when cryptocurrency boom took place.

protection/antivirus software vendors have added crypto miner detection to their products. Antivirus is one of the good things to have on endpoints to try to protect against crypto mining. Although crypto minor authors are constantly changing their techniques to avoid detection at the endpoint, there's a good chance it will be detected.

- **Maintain browser extensions**: Some attackers are using malicious browser extensions or poisoning legitimate extensions to execute crypto mining scripts.
- **Keep web filtering tools up to date**: If there is a web page that is delivering cryptojacking scripts, we should make sure that the users are blocked from accessing it again.

There is one biggest limitation of these methods - all of them are static. All of these Chrome extensions and Antivirus softwares use a static blacklist/filters which contains regex patterns of the URLs which are involved in Cryptojacking. If a new miner comes, it will have to be manually added to the blacklist to block it in the future. Another disadvantage is that it requires significant efforts to get removed from the blacklist since there is no central community/organization which maintains the blacklist. Instead, different extensions and antivirus softwares maintains their own blacklists. This is the biggest motivation in our work.

This report is divided into following sections. First, we will explain the related work done in this domain. Then, we will explain the architecture that we implemented to survey top 250k Alexa websites. Next, we explain the architecture behind collecting the CPU, Memory Usage and Network metrics of a website. After that, we explain the part where we apply machine learning on the data we collected for each website. Finally, we tell the limitations and scope for future work in this field.

RELATED WORK

There has been few studies conducted to determine the threat of web-based cryptojacking. Although browser-based mining is a novel method for mining cryptocurrency, but lately a large number of websites have started to exploit the user's resources for mining activities. Also, falling into traps of these mining activities is not a rare event as a good percentage of websites are involved in mining. The extent of this phenomenon has lead to the need of a counter measure against the websites mining cryptocurrencies.

There has been a study done by Marius Musch, Christian Wressnegger, Martin Johns, and Konrad Rieck (2018) to survey Alexa top 1 million websites which are involved in cryptojacking. They have proposed a 3-phase analysis approach to identify mining scripts and conduct a large-scale study on the prevalence of cryptojacking in the Alexa 1 million websites. In the first phase, they have detected the ”candidate sites” i.e. the websites which likely but not necessarily host a mining script. In second phase, they conducted a significantly prolonged run-time analysis of the candidate's sites, in which the sites receive no external interaction and hence should be idle after the initial rendering and set-up in legitimate cases. However, if once the page is loaded, all JavaScript is initialized, and the DOM is rendered, the CPU usage still remains on a high level and the computation load is the result of repetitive execution of a single function within the webpage's code base, they concluded that the site hosts an active cryptojacking script. Finally, they extracted the javascript code from the validated mining sites that is responsible for initiating and conducting the mining operations. From this script code, we take both the URL and a hash of its contents as two separate features. Furthermore, we collect all parsed WebAssembly functions, sort them and use the hash of the whole code base as the third feature. We then apply each feature to our list of confirmed miners from the previous phase and keep only those that describe at least a certain number of miners. As the result, we obtain a set of generalized fingerprints, which can identify common mining scripts even in their inactive state.

Apart from the survey, they also performed several secondary analyses to gain insight into the cryptojacking landscape, including a measurement of code characteristics, an estimate of expected mining revenue, and an evaluation of current blacklist-based countermeasures. Apart from the survey, the objective of our paper is completely different. Also, the methodology that we have used perform the survey is also different.

But, unfortunately the current detection and prevention mechanisms are insufficient to tackle this threat as they rely greatly on simple blacklists i.e. extensive lists of the crypto-jacking websites.

III ARCHITECTURE AND METHODOLOGY

This section is divided into three parts. First is the *Survey*, where we explain how we gathered the data of which all websites are involved in cryptojacking. Second is *Gathering System Metrics*, where we collected and analyzed system information like CPU Usage, Netowrk Usage and memory on opening each mining website and 1000 non-mining websites. The last part contains our approach while selecting features and applying machine learning features on the data. Based on the output of Machine Learning model, we then create a Chrome Extension viz. ”George” to predict mining in real time.

1) Survey

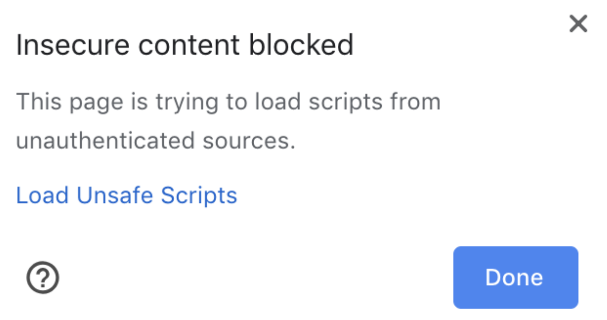
To begin with, we performed a survey on Alexa Top 250k websites to check how many websites are mining cryptocurrency on their user's system. Below is the environment we used to perform the survey:

- **OS** MacOS 10.14
- **System Memory** 8GB
- **Browser** Firefox Quantum: Developer Edition
- **Back-end Server** Heroku PaaS
- **Database** PostgreSQL

To perform this, we used Firefox Developer Edition as the browser. Below are the reasons for us to choose Firefox Developer Edition web browser over Chrome and normal Firefox web browser.

- Chrome extension has a default functionality to block ”insecure” content on a website. This insecure content includes unsecure javascript code. This javascript code sometimes contain code to mine the cryptocurrency. For example, in figure 2, the warning comes when opening a website with ”unsafe” scripts.

- Firefox Developer Edition allows permanently installing a browser extension which has not been published yet. Since we used a ”customized” Minerblock extension which we specially modified as per our requirements, we chose to use this browser.


Insecure content blocked

This page is trying to load scripts from unauthenticated sources.

Fig. 2: The above image shows the warning in Chrome browser on opening website URL https://kat.sx, which is one of the miner websites.

Next, we developed a webservice and deployed it on *Heroku*, which is a Platform as a Service solution and provides server to host any web application. We created a Java based web service and PostgreSQL database to store details of mining websites. Below are the web REST APIs we developed in our webservice.

- **saveMinerWebsite** This *POST* service saves the detail of the miner website in the database. The incoming object has website URL, count of miner scripts, URL of miner scripts and timestamp properties.

- **getMiningWebsitesCount** This *GET* service returns the count of mining websites which are currently stored in the database.

- **getMinerDetailsByURL** This *POST* service returns the details of the website by website URL. The details include the miner script URL, timestamp and the miner count.

- **getAllWebsites** This *GET* service returns all the websites' details which are stored in the database.

Once the web-service is deployed, our next goal was to configure this webservice in a browser extension which will detect if the website in the current tab is Cryptojacking. For this, we chose *Minerblock* [6], which is an efficient browser extension that focuses on blocking browser-based cryptocurrency miners all over the web. The extension uses two different approaches to block miners. The first one is based on blocking requests/scripts loaded from a blacklist, this is the traditional approach adopted by most ad-blockers and other mining blockers. The other approach which makes MinerBlock more efficient against cryptojacking is detecting potential mining behaviour inside loaded scripts and kills them immediately. This makes the extension able to block inline scripts as well as miners running through proxies. This extension is available for Chrome, Firefox and Opera web browser. Although the code for the Chrome version of the web extension is public, it was not public for Firefox browser. On our request [7], the owner of the extension made the code for Firefox extension public.

Once we had the extension's code, we modified it as per our requirement. Figure 3 shows our architecture we used to perform the survey. To start with, we wrote a shell script which will open website URLs from list containing Alexa top 250,000 websites. We configured the shell script to open 25 tabs of the browser at once to make this whole process faster. Also, we kept each tab open for 30 seconds because of below reasons:

- Most of the miner websites do not start mining as soon as they are opened. They wait for at least 5-10 seconds before starting mining activity.
- Since we are opening 25 website at parallel at once, we needed to make sure that we are giving each website enough time to load their DOMs properly.

After opening each of these websites, if there is any miner code on the website's homepage and the same is

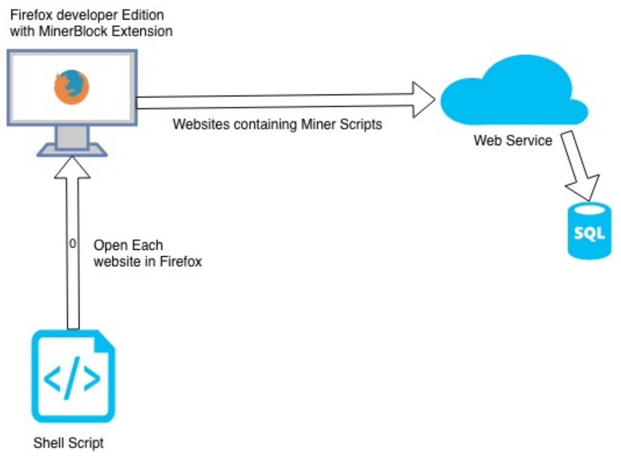

Firefox developer Edition
with MinerBlock Extension
Websites containing Miner Scripts
Web Service
SQL
0
Open Each
website in Firefox
Shell Script

Fig. 3: Architecture to perform the survey of cryptojacking websites

detected by the *MinerBlock* extension, the extension calls the *saveMinerWebsite* endpoint of our web-service with website URL, the miner javascript file name and the number of miner URLs on that same website (mostly it is one). The web-service will then store all these details in our PostgreSQL database.

We ran the whole process on two systems and it took almost 10 days for the complete run.

2) Gathering System Metrics

Once we were done with our survey, we had data for all the website among the top 250k Alexa website through our customized miner block extension that were cryptomining. Now our task was to determine some property through which we would be able to distinguish between these mining websites and non-mining websites.

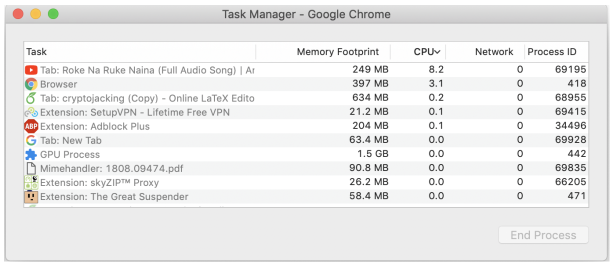
One visible feature that distinguishes miner websites from non-miner websites is that whenever we open a miner websites, the system starts heating up along with unusually high CPU usage. Also, for the purpose of our analysis, we wanted to gather the network usage and memory usage whenever each website is opened.

We tried below ways to gather these metrics.

- **psutil** It (process and system utilities) is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system monitoring, profiling and limiting process resources and management of running processes.[8]

- **systeminformation** It is a lightweight collection of 35+ functions to retrieve detailed hardware, system and OS information in NodeJS. It gives detailed information about system, CPU, baseboard, battery, memory, disks/filesystem, network, docker, software, services and processes. It supports Linux, macOS, partial Windows, FreeBSD and SunOS.[9]

- **chrome experimental APIs** The processes API allows access to information about Google Chromes process model, including process IDs and the CPU usage, network usage and memory consumed by each individual tabs [10]. As we can see in figure 4, the chrome task manager shows different metrics for each currently opened tab.



Task Manager - Google Chrome

Task	Memory Footprint	CPU⌄	Network	Process ID
Tab: Roke Na Ruke Naina (Full Audio Song) | Ar	249 MB	8.2	0	69195
Browser	397 MB	3.1	0	418
Tab: cryptojacking (Copy) - Online LaTeX Edito	634 MB	0.2	0	68955
Extension: SetupVPN - Lifetime Free VPN	21.2 MB	0.1	0	69415
Extension: Adblock Plus	204 MB	0.1	0	34496
Tab: New Tab	63.4 MB	0.0	0	69928
GPU Process	1.5 GB	0.0	0	442
Mimehandler: 1808.09474.pdf	90.8 MB	0.0	0	69835
Extension: skyZIP™ Proxy	26.2 MB	0.0	0	66205
Extension: The Great Suspender	58.4 MB	0.0	0	471

Fig. 4: The above image shows the Chrome's "Task Manager" which contains CPU Usage, Memory footprint and current Network Snapshot of each tab opened.

For our project, we decided to move ahead with the third option i.e. Chrome Experimental APIs because of the following reasons:

- The chrome experimental APIs gives tab-wise metrics. So, if there are some other processes running on the machine which are consuming high CPU or Memory, the readings for our tab in action will not be affected.
- We can open several websites at once and collect their metrics parallelly. This will help in gathering our data faster.

After exploration, we decided to move forward with chrome experimental API which gives the system usage information for any particular tab. This information included three properties of underlying system:

- **CPU** The most recent measurement of the process's CPU usage

- **Network** The most recent measurement of the process's network usage

- **Memory** The most recent measurement of the process's memory usage

To collect these metrics, we developed another chrome extension. The architecture of this exercise is depicted in figure **??** This extension will open all the mining websites and any randomly selected 1000 non-mining websites. For each website, the extension track the above three metrics and takes 120 readings before dumping it into a file. Just as a precautionary measure, we put sleep of 30 seconds to let the CPU return to normal state after dumping the data. This process took almost a day to gather the readings for all the websites.

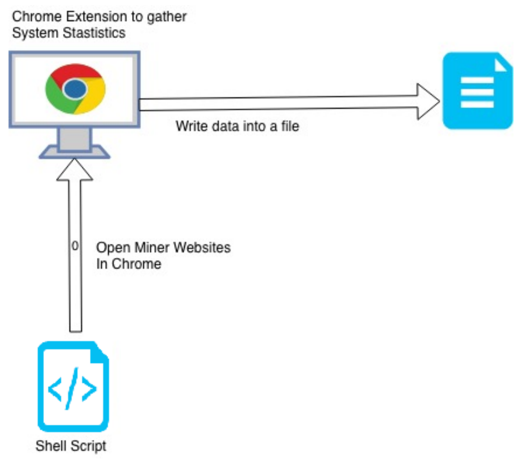

Chrome Extension to gather System Stastistics
Write data into a file
Open Miner Websites In Chrome
Shell Script

Fig. 5: Architecture to perform the metrics collection of miner and non-miner websites

For the non-mining website, we automated the process by listing all the 1000 websites into a file and the extension read the URLs one by one and saved the CPU usage information for all of them. As mentioned in Section III that Chrome might block the miner scripts which can affect our data collection, we opened the 120 mining websites manually in Chrome and made sure that Chrome is not blocking any unsafe scripts from that website. There is a flag in Chrome i.e. *-allow-running-insecure-content* which, as per the Chrome documentation, loads insecure content by default, but this flag was not working on MacOS. The next step in the process was to analyze the data that we collected and import the Json files for all the recorded websites into a dataframe in python code, select features and apply Machine Learning algorithm.

3) Applying Machine Learning

To apply Machine Learning, we first needed to select features to apply Machine Learning algorithm. For that, we analyzed CPU Usage, Memory and Network Usage of miner and non miner websites. Figure 6 depicts the CPU usage for a sample of mining and non-mining website monitored over 2 minutes time. We can clearly see that the the CPU Usage of mining website is consistently higher than that of non mining website. The other two metrics were not reliable enough to distinguish between mining and non mining websites. For example, in Figure 7, we can see that the memory usage for non mining website is lower than that of mining website. Also, this is not always the case for all the websites, so we did not choose this feature for training our model. For the third metric i.e. Network Usage, we found that for most of the websites including mining websites, the figures were negligible values, which is the reason why we chose to ignore this for the training purposes.

Now, we cannot just use single feature to train our Machine Learning model as it will lead to poor predictions.

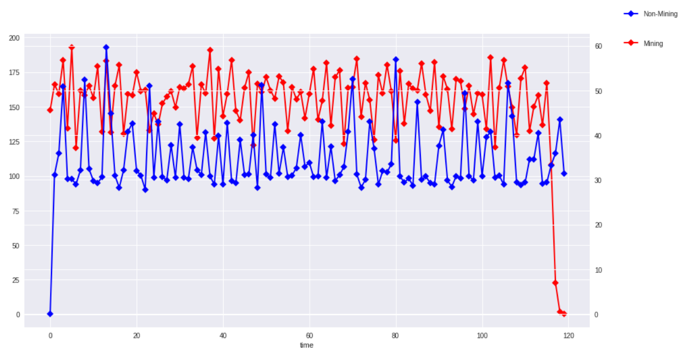


Fig. 6: CPU usage readings for Mining and Non-mining site

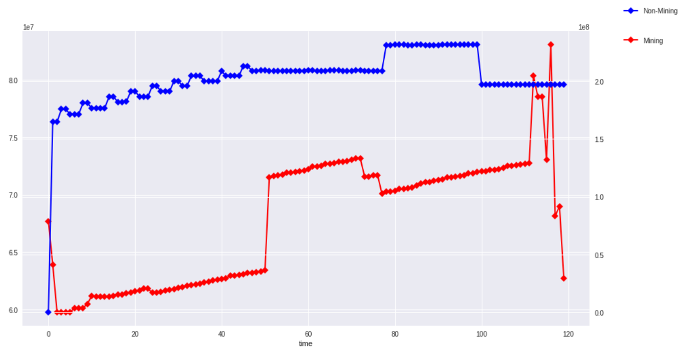


Fig. 7: CPU usage readings for Mining and Non-mining site

To overcome this, we converted our CPU Usage for each website into 8 different features. Since we gathered the metrics for each website for the period of 2 minutes, we divided this data into 8 sets of 15 seconds each. We calculated the mean of each 15 seconds interval and then use each of those values as a different feature.

Once we have the data and required features, we needed to predict if a website being browsed by user is involved in mining activity or not. We decided to use a supervised machine learning algorithm viz. *Logistic Regression*, a classification model for clustering the websites into two clusters/categories : *mining* websites and *non-mining* websites. Logistic Regression also assigns a probability score for the website to lie in each of the clusters. This module takes the features associated to a record and the true category of the record as the inputs. Next step was to formulate the data to the format which is accepted by the model. All the reading values for the CPU usage where formulated in a matrix, say X and all the true categories of these records formed the matrix Y. These two matrices we feed into the training model.

We have used in total 1,100 websites to train the logistic regression model out of which 100 were mining websites and rest non-mining websites. One major operation performed on this collected data before feeding it to the machine learning model was re-sampling to reduce the bias in the training data towards one category. This is because we the number of mining websites were comparatively lesser than non mining websites. So, we created duplicate records of mining websites data to equalize the data for mining and non mining websites.

The next step was training and testing of our logistic regression model. After re-sampling we had 1000 cryptomining and 1000 non-cryptomining websites to be fetched into the model. Once the model was trained we used it to evaluate its performance by running it on 100 test data websites. The algorithm performed astonishingly well with an accuracy of 98%.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Fig. 8: Formula to calculate the probability of a website being a mining website

The coefficients of the model after training were later leveraged to perform real-time detection of the cryptomining sites through the web extension ”George”. This extension keeps accumulating the CPU usage data for each of the opened tabs for each 2 minute intervals for the first 10 minutes. After first 2 minutes, we calculate the mean of 15 seconds intervals similar to what we derived for the training purposes. Once we have the features, we used the formula in Figure 8, to calculate the probability of website being a mining website. We keep repeating the above process for first 10 minutes.

IV RESULTS

We carried out a survey for the Alexa top 250,000 websites to evaluate if they were indulged in cryptomining. We found that 120 websites to be mining cryptocurrency which is 0.05% of the total sites under observation. Also, due to the recent crash in Cryprocurrency prices, it might be possible that the number of mining websites have come down.

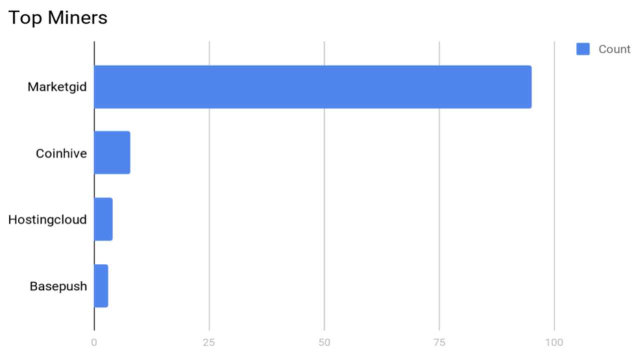
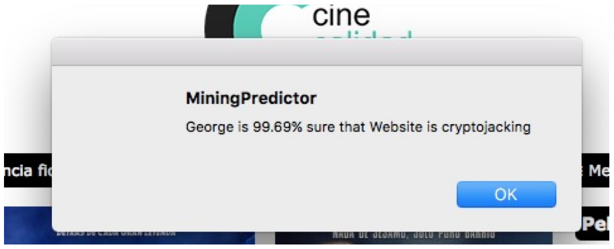


Fig. 9: The above image shows the count of miner scripts employed by most number of miner websites

We predicted for 100 random sites if they were mining cryptocurrency or not and obtained an accuracy of 98%. Figure 10 shows our extension ”George” in action in detecting the cryptomining website. There were interesting insights that we gathered on the False positives and False negatives that we encountered on the test data of 100 websites.

- **False positives:** There were small number of false positives in our results. These occurred when we were collecting the data for Video Streaming websites and constantly switching between two pages, this usually leads to high CPU Usage.
- **False negatives:** Our algorithm produced no false negative, that is no website that was mining cryptocurrency was marked as a non-mining website. This solves our purpose that no website that is involved in cryptomining activities should go undetected by our model.



MiningPredictor

George is 99.69% sure that Website is cryptojacking

Fig. 10: Snapshot of George detecting a cryptomining site

After in depth analysis, we found that Marketgid script is responsible for most of the mining taking place. We were able to create an extension which in real time predicted if a website being surfed by user is a mining website or not. The accuracy of the real-time extension was very impressive when tested against our test data. The system after being on a website for about 2 minutes could successfully categorize it as a mining website.

V LIMITATIONS

Our final model provided us results of predicting the mining sites with great accuracy. But there are still some improvements and limitations associated with the model. Firstly, while collecting the data for the mining websites we tested only the homepage of a website to conclude if a website is a mining website or not. But it is a very common behaviour with the websites that they have mining scripts running and embedded on the navigation pages other than the landing pages.

Another factor to be taken into notice is that coefficients obtained from logistic regression training that we use to predict the mining websites in real time is static. So there would be frequent need of training the model again to obtain the updated coefficients as the nature of websites and the cryptomining around us changes.

As a part of our analysis we could not record data for certain websites as they were blocked in the Stony Brook network. We used a static blacklist to check for the miner websites. Since this list is static and does not update frequently, many new mining websites would not be detected when compared against this blacklist. Our study involved opening and observing a website for 30 seconds to conclude if it is using the mining scripts, this system lacks the inclusion of the websites that starts mining activity after this time frame.

VI FUTURE WORK

We have used a fixed set of Alex top 250,000 websites for training the machine learning algorithm. As future work, we would include more websites as the input with more diversity. One more future improvement to the system is to reduce the time for the real-time detection. As we know our machine learning model required an input of 2 minutes of CPU usage against a website, so the extension displays the prediction after a lag of 2 minutes. We hope to improve this lag time as a future improvement.

Our current tool ”George” only reports the user if a website being browsed is involved in mining or not. One of the key feature that we aim to add is the Blocking functionality, that is blocking the website altogether once the system categorizes it as a potential miner website. Our machine learning model for now takes CPU usage for 2 minutes interval of time as the input for classification and categorization. We look forward to determine more such meaningful factors and systems features which could lead to better prediction accuracy of the system.

As we have used experimental chrome API, so right now we were not able to publish the chrome extension to chrome web store. Hopefully this experimental API will be available for normal usage and we will try to publish our extension.

VII CONCLUSION

As part of this project, we were able to accomplish several things. We started with surveying top 250,000 websites to check how many of them are mining. After that, we derived the system metrics like CPU Usage, Network Usage and Memory Footprint for each website when opened in a web browser. We tried several ways to collect the above metrics, but finally zeroed down to using Chrome experimental APIs due to several advantages listed in Section III. Using the above metrics, we trained a Logistic Regression Model and developed ”George”, a smart Chrome Extension to detect Cryptojacking in real time. This is the first time a dynamic way is developed to detect cryptojacking.

VIII CONTRIBUTIONS

The first part of our project that dealt with conducting the survey of the Alexa top 250K websites was lead by Rishabh Jain. He also performed customization of the MinerBlock extension to accommodate our requirements of calling the web service which stored the information for the mining sites.

Aman Raj continued with the task of collecting the CPU usages of the mining and non-mining websites. For achieving this task, he developed a Chrome extension which used experimental chrome API for logging the CPU usages. He did extract the Json files for all the websites which were later fed as the input to the machine learning model.

The development of the classification model to predict and classify an input website to be a mining webisite or not was handled by Digant Jain. He used logistic regression model as the classification technique which also provided the probabilities of each website to be a mining website or not. The coefficients generated by training the model were later leveraged in the extension to predict the cryptomining websites in real time.

REFERENCES

[1] https://arxiv.org/pdf/1808.09474.pdf

[2] S. T. Ali, D. Clarke, and P. McCorry. Bitcoin: Perils of an unregulated global p2p currency. In Security Protocols XXIII, pages 283-293. Springer, 2015.

[3] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In Proc. of IEEE Symposium on Security and Privacy, pages 104-121, 2015.

[4] https://www.investopedia.com/terms/c/cryptojacking.asp

[5] https://www.csoonline.com/article/3253572/internet/what-is-cryptojacking-how-to-prevent-detect-and-recover-from-it.html

[6] https://github.com/xd4rker/MinerBlock

[7] https://github.com/xd4rker/MinerBlock/issues/34

[8] https://pypi.org/project/psutil/

[9] https://www.npmjs.com/package/systeminformation

[10] https://blog.chromium.org/2010/03/experimental-extension-apis.html