Detecting malicious campaigns in obfuscated JavaScript with scalable behavioral analysis

1st Oleksii Starov
Palo Alto Networks, Inc.
ostarov@paloaltonetworks.com

2nd Yuchen Zhou
Palo Alto Networks, Inc.
yzhou@paloaltonetworks.com

3rd Jun Wang
Palo Alto Networks, Inc.
junwang@paloaltonetworks.com

***Abstract*—Modern security crawlers and firewall solutions have to analyze millions of websites on a daily basis, and significantly more JavaScript samples. At the same time, fast static approaches, such as file signatures and hash matching, often are not enough to detect advanced malicious campaigns, i.e., obfuscated, packed, or randomized scripts. As such, low-overhead yet efficient dynamic analysis is required.**

In the current paper we describe behavioral analysis after executing all the scripts on web pages, similarly to how real browsers do. Then, we apply light "behavioral signatures" to the collected dynamic indicators, such as global variables declared during runtime, popup messages shown to the user, established WebSocket connections. Using this scalable method for a month, we enhanced the coverage of a commercial URL filtering product by finding additional 8,712 URLs with intrusive coin miners. We evaluated the impact of increased coverage through telemetry data and discovered that customers attempted to visit these abusive sites more than a million times. Moreover, we captured 4,633 additional distinct URLs that lead to scam, clickjacking, phishing, and other kinds of malicious JavaScript.

Our findings present up-to-date trends in unauthorized cryptographic coin-mining, and show that various scam kits make up another big fraction of the modern threat landscape on the Web.

I. Introduction

Fine-grained URL categorization is an important and integral service to enterprise security solutions in that it offers protections to customers from a myriad of web-based threats, such as exploit kits, malicious redirections, phishing pages, scams, and potential inappropriate content (e.g. adult, weapons, drugs) for sensitive audience groups. As the web scales exponentially every day both in terms of quantity and complexity, categorizing the web quickly and accurately is becoming a more and more daunting yet important task. Identifying the category of a URL usually involves crawling its content, analyze, and then make a decision. An automated system has to do this millions of times on a daily basis due to the scale of the wild web. To complicate things, dynamic web content based on potentially obfuscated JavaScript creates even more hurdles for automated systems to identify its category correctly. Recently, unauthorized cryptographic coin-mining using JavaScript is an excellent example of this web dynamism. Due to this, fast static approaches to search for malicious scripts, such as file signatures and hash matching, are often insufficient in detecting advanced malicious campaigns, i.e., obfuscated, packed, or randomized scripts. To detect such advanced cases on the given large-scale, low-overhead yet efficient dynamic analysis is required.

There are many previous researches which enrich JavaScript classifiers with dynamic features or build behavioral models for malicious scripts [2], [9], [12]. However, proposed systems usually require additional instrumentation and resources that are unsuitable for near real-time analysis of large population of URLs. Moreover, such approaches use machine learning models, which must be trained and calibrated to minimize false positives. In contrast, there is lack of research in terms of *behavioral* detection methods, i.e. indicators of dynamic script execution that conclusively determine malicious behavior. To the best of our knowledge, we are the first to present a systematic case-study on detecting modern malicious campaigns with such scalable dynamic analysis.

In this paper, we describe behavioral analysis that executes all the scripts on web pages, similarly to how real browsers do, and then observes their behavior. We apply "*behavioral signatures*" to the collected dynamic information, such as global variables declared during runtime, popup messages shown to the user, and established WebSocket connections/messages sent. In the end, if we find a confident match with behavior of a known malicious family, we flag the corresponding page as corresponding category. By using such dynamic behavioral analysis with a low performance overhead, we can effectively detect obfuscated malicious code as their final malicious behavior remains unchanged.

Applying this detection on URLs collected by a commercial security solution provider over one month, we enhanced the coverage of its URL categorization service by finding additional 8,712 URLs with intrusive cryptographic coin miners. Telemetry data collected by this provider shows that its customer attempted to visit these abusive sites more than one million times. We show that intrusive coin miners continue to be one of the main threats on the Web, and that the majority of mining happens without user's consent even on top popular websites. In addition to coin-mining websites, we captured other highly-popular 4,633 distinct URLs that lead to scam, clickjacking, phishing, and other kinds of malicious JavaScript.

II. Behavioral Signatures

There are many ways how malware can manifest its malicious behavior after execution. Malicious JavaScripts span var-

```
var miner = new CoinHive.User('SITE_KEY', 'john-doe');
miner.start();

var miner = new CRLT.Anonymous('PUBLIC_KEY', {threads:2});
miner.start();
```

Listing 1: Examples of coin-mining integration scripts.

```
(function(){
    var _0xdf51=["\x70\x61\x72\x61\x6D\x73","\x5F\x73\x69\
    x74\x65\x4B\x65\x79","\x5F\x75\x73\x65\x72","\x5F\x74\
    x68\x72\x65\x61\x64\x73","\x5F\x68\x61\x73\x68\x65\x73
    ","\x5F\x63\x75\x72\x72\x65\x6E\x74\x4A\x6F\x62","\x5F
    \x61\x75\x74\x6F\x52\x65\x63\x6F\x6E\x6E\x65\x63\x74",
...
MINER_URL:_0xdf51[207],AUTH_URL:_0xdf51[208]};CoinHive[
_0xdf51[104]]= CoinHive.Res(_0xdf51[209]);var user=window[
_0xdf51[211]][_0xdf51[210]]|| _0xdf51[212],miner= new
CoinHive.User(_0xdf51[213],user,{throttle:0.3});miner[
_0xdf51[89]]()|| miner[_0xdf51[53]]()
})();
```

Listing 2: Example of an obfuscated coin miner.

```
var hea2p = ('0123456789
ABCDEFGHIJKLMNOPQRSTUVXYZabcdefghijklmnopqrstuvxyz');
var hea2t =
'UQIqIxFiqFvTwP5ovMqrc9ml0GHxUqsv0qWaSdrXOFJw3VOAJf1gvPk+
QBh7Vp+Mw6Z6uS+kTPYikXQcCQtwsrY9Q4bxTlp19uQHk6poa1mGy8/1
uCIrMarTmM01aa0CR7yOoa3nPkG1wmorbr14ETB+
pWCWfV737Wr03qjYFQmxrhe7k7+EbDDZtW+OdbGci6S2Uqmg82Y
...
SqorG0GsD0gEcRIOpupIBe1s8csW1EbpOcMBBi2YOsszLVtU9sqoLFvqME
+0UZS+nIGpc7UgjPloOrRRTWne+so1j55dTzvE9jABmqgGxOB6PGs=';
var output = Aes.Ctr.decrypt(hea2t, hea2p, 256);
document.write(output)
```

Listing 3: Example of a phishing kit.

```
...
function ClickJackFbHide() {
  jQuery("div[idˆ='clickjack-button-wrapper']").hide();
}
function ClickJackFbShow() {
  jQuery("div[idˆ='clickjack-button-wrapper']").show();
}
...
```

Listing 4: Example of a clickjacking kit.

ious families which perform totally different harmful actions (e.g., coin miners establish WebSocket connections to transmit mining data, scareware pages show frightening alerts, phishing kits mimic popular websites). However, most malicious JavaScripts we observe create variables, functions and other execution artifacts, and as we show below, such indicators like variable names declared in a global scope are often unique and descriptive enough to create a high-confidence signature. Moreover, in case of obfuscated JavaScripts, we observe that the same discriminating variables are often revealed during execution as malware simply unpacks and evaluates (e.g. via "eval" or "document.write") the original malicious code. This fact makes dynamic retrieval of variable names an important source of behavioral signatures, and so we will describe it first.

A. Global Variable Signatures

In order to illustrate this approach, let's take a look at some snippets of a popular type of malicious JavaScripts

Unauthorized cryptographic coin miners. Popular coin-mining libraries like *Coinhive* and *Crypto-Loot* usually require standard snippets on a web page to start mining, such as shown on Listing 1. As we see, in the first case, *CoinHive* is visible as a global variable on the page, and in the second case, similarly, the Crypto-Loot's *CRLT* variable is visible. In addition, in both cases the code creates the variable *miner* which is accessible via window object. As a result, we can easily look for the existence of such variables after rendering the web page.

Though, the main benefit of the *dynamic execution* is explained by the following example from a real website. While previous code snippets can be detected by other methods such as static signatures, or just by the presence of a coin-mining library on the page (*coinhive.min.js* or *crypta.js* correspondingly), the whole coin-mining code can be obfuscated in a way as shown on Listing 2.

In such obfuscated cases, matching static signatures and file hashes face obvious difficulties, but when we execute such scripts – it is straightforward to detect the same known global variables created on the page, namely *CoinHive* and *miner*. In this study, we encountered many examples of the mining JavaScript libraries similar to the above, served from custom servers and usually named unrecognizably or even masked as popular benign scripts, such as *jquery.min.js*. Due to the fact that most of these coin miner libraries have a larger-sized library and smaller bootstrap snippet for sites to customize miner ID and/or mining parameters such as throttle and number of cores used, the larger library cannot wrap itself in a closure, i.e. has to expose global variables for snippet to refer. This gives us excellent opportunity to discover miner presence by iterating JavaScript variables under "window" object. To detect results presented in this paper, our system is able to detect at least 23 popular coin-mining families by global variable signatures.

Phishing kits. In this study, we observed many malicious scripts with similar structure as on Listing 3. While unpacking its phishing payload, the script creates several global variables, namely *hea2p*, *hea2t* and *output*. In combination these variables create a high entropy signature, which can be used to facilitate detection.

Clickjacking kits. Another type of malicious JavaScript campaign discovered on many websites during November 2018 is the clickjacking campaign that targets at Facebook's Like clicks. Its code, partially shown in Listing 4, creates several rather descriptive global functions, including *ClickJackFbHide* and *ClickJackFbShow*.

In addition, specific cases like redirectors, iframe droppers and even particular JavaScript exploits can be identified by variable signatures derived with dynamic execution.

B. Malicious Websocket Connection Signatures

While presence of variables can be a strong enough signal in some cases, we sometimes need additional proof of the

```
{
    "type": "submit",
    "params": {
        "version": 7,
        "job_id": "871932594873942",
        "nonce": "8a462f80",
        "result": "7516e787d860071b24961aca...de4df27f300"
    }
}

{
    "type": "hash_accepted",
    "params": {
        "hashes": 256
    }
}
```

Listing 5: Examples of Startum request and response.

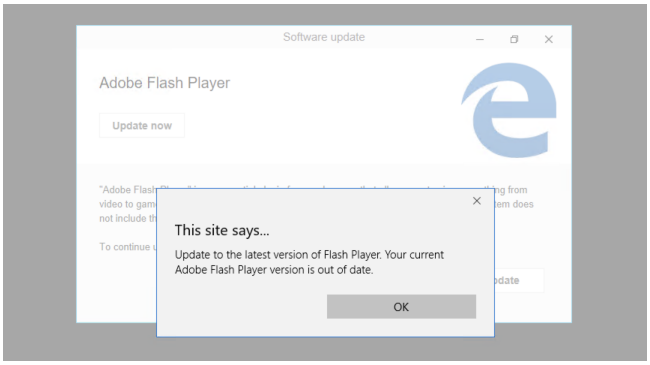

Fig. 1. Example of a Fake Flash Update Scam.

actual *mining behavior that starts without user's consent*. To this end, we use additional methods to detect the start of the mining process.

The malicious party will never benefit from injected coin-miners if it does not receive result back from the browser client. Coin-mining JavaScripts do so by establishing WebSocket connections and exchanging messages between client and server. This gives us the ability to implement detection signatures with high confidence, since most mining libraries rely on the standard Stratum Protocol [1], which has detectable patterns. For example, Listing 5 illustrates messages exchanged during in-browser *Coinhive* mining via WebSocket connection.

Even when mining code encrypts or obfuscates its WebSocket traffic, which happens in case of more sophisticated Stratum wrappers, it is often still possible to derive specific per-miner signatures as obfuscation is not randomized. For example, the CoinImp miner, which we found to be a popular choice for unauthorized coin-mining, obfuscates its WebSocket traffic. However, its messages still have unique patterns (such as, "*suu9sLms6/*" prefix), which are constant across all the websites that use this library. In the result, we can effectively detect mining with *CoinImp* by looking for specific WebSocket messages.

C. Malicious intrusive API calls

Another type of highly popular malicious JavaScript, namely *scam* and *scareware* campaigns, often show alert messages to trick visitors. For example, Figure 1 shows screenshot of a common fake Flash update scam. Pretending to be legitimate warnings, such web pages trick users into installing malware. Such campaigns periodically rotate their templates, text content, and usually adjust the overall look to match a victim's browser and operating system. However, they continue to show JavaScript popups with unique scam messages. As the result, such messages become effective malicious indicators. To detect scareware campaigns, we intercept all the popups that a website shows to a visitor via JavaScript APIs (namely, *window.alert*, *window.prompt* and *window.confirm*), and compare the captured content with prepared signatures for different kinds of scam messages. Note that this approach is also effective against obfuscated alert messages as the attacker must unfold however many layers of obfuscation to display the original message.

III. Data Collection

In this paper, we report on the detection results using behavioral signatures for around one month (from October 19th to November 19th, 2018). In particular, we measure how behavioral analysis can improve a state-of-the-art commercial URL categorization service. For that, we leverage an existing crawling and analysis infrastructure of Palo Alto Networks, a cybersecurity company, which provides its customers with URL categories, such as malware, phishing, or benign. We applied behavioral analysis to suspicious URLs visited by Palo Alto Networks customers. In addition, every day we crawled Alexa's top one million websites, as well as a feed of newly registered domains.

The crawler infrastructure we use is based on the headless Chrome browser. Our crawler is capable of collecting global variables, intercepting JavaScript alerts, and recording WebSocket handshakes and messages. For each test, we explicitly wait for additional 3 seconds after page load in order to capture potentially delayed on-page behavior. With such short yet effective delay, our crawler infrastructure can process over 10 million URLs per day.

Overall, we used more than 250 unique signatures for detection, based on global variables, JavaScript alerts, WebSocket URLs and messages. While this set of behavioral signatures is far from being exhaustive (as well as must be updated regularly), our results show a lower bound on the possible detection rate. This set of signatures was retrieved automatically from malicious websites known to VirusTotal, as well as checked and added manually.

Finally, our passive DNS database used in Section IV-C consists partially of our customer's DNS resolution as well as passive DNS data provided by Farsight Security.

IV. Analysis of Results

During a month of crawling, the proposed approach detected an additional 9,104 coin-mining scripts (among 8,712 distinct URLs), and 4,788 malicious JavaScripts (among 4,633 distinct

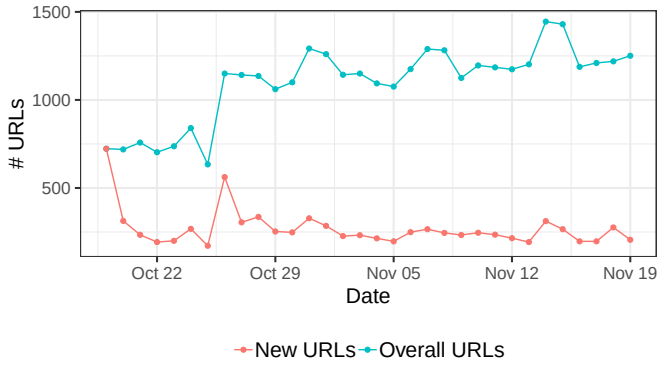

Fig. 2. Daily Detection Rate of Coin-mining URLs.

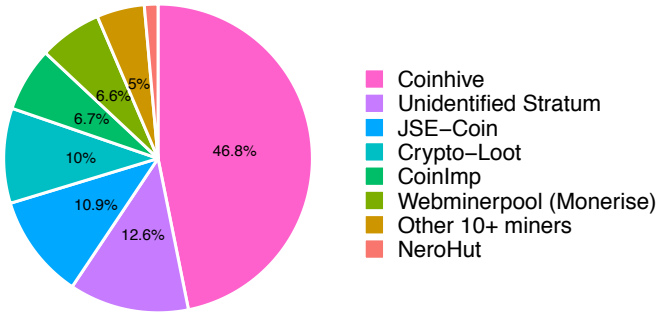

Fig. 3. Top Recently Observed Coin Miners.

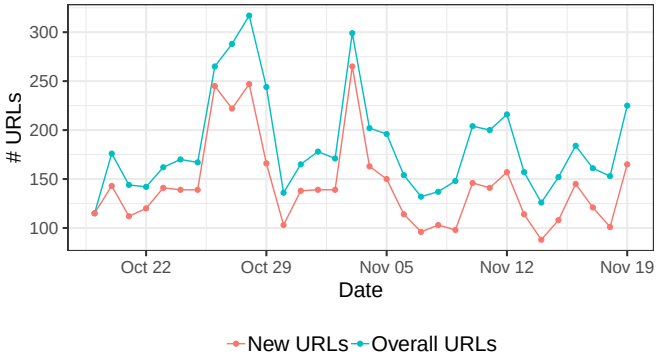

Fig. 4. Daily Detection Rate of Scam URLs.

URLs). The impact of these malicious campaigns on real web users is extensive, as according to our telemetry data, our customers have attempted to make over 1 million HTTP(S) requests towards the detected coin-mining websites. As for scams/scarewares/phishing pages, there were over 243 thousand requests made. We present the details of our discoveries in this section.

A. *Coin-mining Scripts*

By using behavioral JavaScript analysis alone, we detected 1,097 coin-mining URLs per day on average, including over 180, previously unseen, malicious URLs per day. Figure 2 shows the daily detection rates. As we can see, despite periodical bumps, the trend on newer detections remains stable. The fact that we are detecting the same coin miners from day to day highlights the fact that 1) coin-mining websites are longer living in comparison to other type of malicious JavaScripts; 2) Our customers are visiting those sites frequently due to their popularity.

Figure 3 illustrates the distribution of popular families of coin miners among 9,104 scripts detected during the month. As we see, *Coinhive* is leading with 46.8% of the share, and followed by 12.6% of anonymous Stratum Protocol communications, as well as other coin-mining libraries, such as *JSE-Coin* (10.9%) and *Crypto-Loot* (10%). More than 10 other coin-mining libraries, which include *CoinNebula, BatMine, DeepMiner, CryptoNoter, Minr* and others, together contribute to only 5% of all the detected scripts, indicating a longer tail in the distribution of coin miners.

Overall, 6,026 out of 8,712 URLs with coin miners, or 69.1%, perform "*unauthorized*" coin mining. In other words, they start mining immediately after visiting the page without user's consent. The most popular choice among coin-mining libraries to perform unauthorized mining were CoinImp (in 100% cases over 612 scripts), Crypto-Loot (in 86% cases over 906), Coinhive (in 77.7% cases over 3,312), and other 1,143 scripts generating unidentified stratum traffic.

In addition, we observe that many coin mining campaigns deploy techniques to remain stealthy, which mostly includes static obfuscation and packing of their JavaScript. For example, at least 1,414 out of 4,264 Coinhive scripts, or 33.2%, were detected only because we were using dynamic analysis with behavioral signatures, as other methods, such as hash or JavaScript AST (abstract syntax tree) matching could not detect the obfuscation. At the same time, we detected many cases when the same web page rotates coin miners from visit to visit, or even hosts several different coin-mining libraries at the same time. For examples, such pairs as Coinhive and JSE-Coin, or Crypto-Loot and NeroHut. Because of this fact, the overall number of detected mining scripts for one month (9,104) was greater than the overall number of unique URLs (8,712).

B. *Other malicious scripts*

Apart from coin miners, we were able to detect on average 184 additional malicious URLs per day, including 143 previously unseen URLs. Figure 4 shows the daily detection rate. As we see, rate for the overall detected URLs and newly discovered URLs are much closer to each other than on Figure 2 for coin miners. This means that URLs hosting scam-related and other malicious JavaScript live shorter and are less likely to be observed for many days in a row. This is expected since the coin miners are often injected into popular but compromised websites, whereas phishing or scam pages are dedicated URLs specifically created for malicious purposes and rotate quickly due to frequent blacklist updates by security vendors.

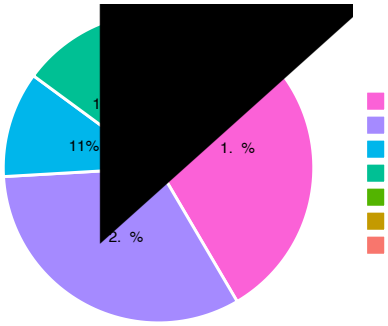

Fig. 5. Top Recently Observed Non-Mining Malicious JS Scripts.

```
"WARNING! Your official Adobe Flash  Player version is out
 of date. Please install latest software update to
continue. Please click \"Update\" to continue."
```

Listing 6: Frequently observed fake Flash scam alert.

Figure 5 presents the distribution of various types of malicious JavaScripts over 4,788 discovered samples. Technical support scams and fake Flash update scams are the most popular malicious campaigns, and others include campaigns like phishing kits, clickjacking kits, and fake reward.

Technical support scam pages [5] trick users into calling and paying scammers, or installing unwanted programs. Our detection reveals that many of such pages likely belong to a single massive campaign operated by the same malicious actor. We observed that 1,668 out of the 1,989 URLs that lead to technical support scam have the same screenshot; they were also showing JavaScript popup with the same text; Listing 6 shows the most popular alert content. Many recent landing pages of these URLs were hosted on domains that followed the same pattern (for example, gaf9342[.]space, gba9462[.]site and gbc2851[.]online). On the other hand, fake flash update scam is much more diverse, showing different pages in cases of Google Chrome vs. Internet Explorer, MS Windows vs. Mac OS.

It is important to emphasize that exact string match of such alerts would result in a limited number of detections - instead we build our signatures based on the most significant terms indicating scam. To this end, we took into account different text randomization and obfuscation tricks that attackers may apply, such as multiple spaces between keywords as with “Flash Player”, invisible characters/new lines, and messages in other languages.

C. Analysis of malicious domains

Overall, the 8,712 coin-mining URLs we detected resolve to 7,654 distinct domains, and 4,633 URLs with other malicious scripts resolve to 2,666 domains. We looked up both sets of domains in our passive DNS (pDNS) database, which gave us ability to estimate both their life span and popularity from a broader view than our customers alone. Figure 6 presents a scatter plot showing for how many days it was active and how many times it was resolved (i.e. popularity) for each domain.

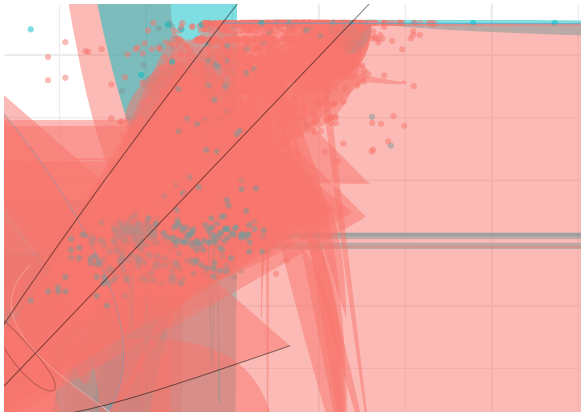

Fig. 6. Lifetime vs. Popularity of Malicious URLs.

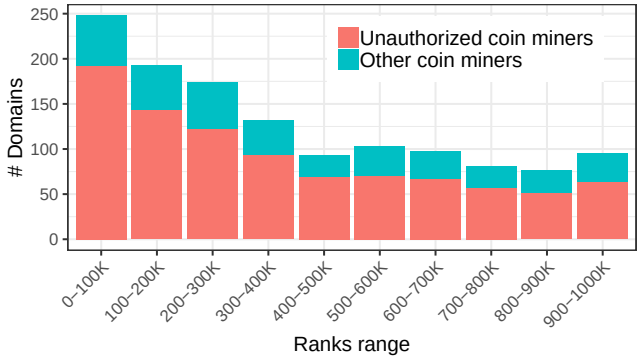

Fig. 7. Rank Distribution of Coin-mining Domains in Alexa’s Top Million.

One may notice that coin-mining domains are significantly more popular and longer-living, whereas other malicious domains are usually alive for less than 100 days and receive less than 10,000 DNS resolutions (lower cluster of blue dots). It can be explained by the fact that scam and phishing campaigns get quickly discovered and blocked by security vendors, and thus have to rotate their domains. Contrastingly, coin miners usually fall into gray zone, being malicious enough that some users want to block the mining, while others may be fine with it as long as web content is served correctly. Among Alexa top websites we have seen many reputable ones that deliberately incorporate coin miners, forcing their visitors to accept that in exchange for “free” content access.

Figure 6 shows that many coin-mining domains are extremely popular, receiving more than tens of millions DNS resolutions and are alive for more than 4 years. These examples include domains directly related to mining (such as, coinhive[.]com, or moonbit[.]co[.]in), and various websites with popular or high-demand content (such as, xxgasm[.]com and seriesfree[.]to). Interestingly, we also see domains with many DNS requests, but with shorter lifetime. We suspect that those are cases when a popular “shady” website registers a new domain, for example, indoxxi[.]cool which serves online

TABLE I
TOP 5 TLDS SERVING COIN MINERS AND OTHER MALICIOUS JAVASCRIPT

Coin-mining websites		Non-mining malicious websites	
TLD	*# Domains*	*TLD*	*# Domains*
com	3,487	icu	1,299
ir	394	com	409
net	337	club	290
ru	319	xyz	83
org	285	tk	61

movies has 63,694 resolutions in pDNS but is only alive for 14 days.

Examples of non-mining malicious JavaScript were usually hosted on dedicated domains, for example, win32-0x2ndt-firewall-error[.]gq or www2[.]betterdealupgradeaflash[.]icu, and in general on less popular websites. One may notice that on Figure 6 there are also rare cases of highly popular and long-living non-mining domains. For example, these are domains of URL shorteners (e.g., ow[.]ly), domains serving ads and frequently used by scam campaigns (e.g., elitedollars[.]com), and popular but rather shady websites (e.g., mytorrents[.]org).

After analyzing Alexa rankings for TLD+1 versions of the detected coin-mining domains, we were surprised to find that 66.9% of them were in Alexa's top million at least for one day during the month of measurements. Moreover, 1,295 of coin-mining domains remained in the top million, including 37 domains in the first 10K. On one hand, these results show that when reputable websites begin to adopt mining, they are likely to drop out from the top popular websites. On the other hand, we are still alarmed by the high number of unauthorized coin miners among top million websites. Figure 7 shows the distribution of 1,295 coin-mining domains across rank ranges, and we clearly see that unauthorized miners dominate even in the top 100K. In contrast, only 71 from non-mining malicious JavaScript domains were found in Alexa's top million, including 7 in top 10K. One of the most popular examples for both mining and non-mining categories of malicious JavaScripts were websites hosted on blogspot[.]com (and having own distinct subdomains).

Finally, Table I lists the most popular TLDs which serve coin miners and other malicious scripts. As we see, .icu TLD was abused the most to distribute different kinds of scam. Regarding coin-mining domains, most of them reside on the reputable .com TLD, which matches our previous finding that miners present on more popular and long living websites (such as blogs, video downloads, torrents).

V. RELATED WORK

To the best of our knowledge, we report the first case study on detecting malicious JavaScript with light-weight yet straightforward dynamic signatures, such as global variables. A variety of previous research works classify malicious scripts solely based on static features of their source code [4], [11], focus only on drive-by-download exploits [2], or develop complex behavioral models [9], [12], which require more instrumentation and resources. Other works develop fine-grained classification of malicious scripts [10], but do not consider newer campaigns, such as in-browser coin miners.

In general, in-browser coin-mining was described in [8], and methods to identify coin miners, including through WebSocket traffic, were described in [3]. In current paper, we present up-to-date statistics on coin-mining websites visited by millions of real-users, and detected by combination of methods. Similarly, technical support scam was previously studied in [6], and social engineering download attacks were measured in [7].

VI. CONCLUSION

In this paper, we presented several examples of light-weight dynamic techniques to detect obfuscated malicious JavaScript. Behavioral signatures based on global variables, alert texts, and WebSocket messages were shown to be effective in detecting modern scam campaigns and unauthorized coin miners. With such behavioral analysis we can also separate unauthorized coin-mining from consented coin-mining by detecting the mining traffic, thus making a stronger case to block such malicious websites.

REFERENCES

[1] "Stratum mining protocol," https://slushpool.com/help/manual/stratum-protocol.

[2] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 281–290.

[3] R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, and G. Vigna, "Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1714–1730.

[4] P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious javascript detection using classification techniques," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, oct. 2009.

[5] N. Miramirkhani, O. Starov, and N. Nikiforakis, "Dial one for scam: Analyzing and detecting technical support scams," *CoRR*, vol. abs/1607.06891, 2016. [Online]. Available: http://arxiv.org/abs/1607.06891

[6] ——, "Dial one for scam: A large-scale analysis of technical support scams," in *NDSS*, 2017.

[7] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, "Towards measuring and mitigating social engineering software download attacks," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 773–789.

[8] J. Rüth, T. Zimmermann, K. Wolsing, and O. Hohlfeld, "Digging into browser-based crypto mining," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: ACM, 2018, pp. 70–76.

[9] K. Schütt, M. Kloft, A. Bikadorov, and K. Rieck, "Early detection of malicious behavior in javascript code," in *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '12. New York, NY, USA: ACM, 2012, pp. 15–24.

[10] J. Wang, Y. Xue, Y. Liu, and T. H. Tan, "Jsdc: A hybrid approach for javascript malware detection and classification," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 109–120.

[11] Y. Wang, W.-d. Cai, and P.-c. Wei, "A deep learning approach for detecting malicious javascript code," *Security and Communication Networks*, vol. 9, pp. n/a–n/a, 02 2016.

[12] Y. Xue, J. Wang, Y. Liu, H. Xiao, J. Sun, and M. Chandramohan, "Detection and classification of malicious javascript via attack behavior modelling," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, ser. ISSTA 2015. New York, NY, USA: ACM, 2015, pp. 48–59.