A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks

Ege Tekiner[§], Abbas Acar[§], and A. Selcuk Uluagac
Cyber-Physical Systems Security Lab
Department of Electrical & Computer Engineering, Florida International University, Miami, FL, USA
{egetekiner, aacar001, suluagac}@fiu.edu

***Abstract*—Recently, cryptojacking malware has become an easy way of reaching and profiting from a large number of victims. Prior works studied the cryptojacking detection systems focusing on both in-browser and host-based cryptojacking malware. However, none of these earlier works investigated different attack configurations and network settings in this context. For example, an attacker with an aggressive profit strategy may increase computational resources to the maximum utilization to benefit more in a short time, while a stealthy attacker may want to stay undetected longer time on the victim's device. The accuracy of the detection mechanism may differ for an aggressive and stealthy attacker. Not only profit strategies, but also the cryptojacking malware type, the victim's device as well as various network settings where the network is fully or partially compromised may play a key role in the performance evaluation of the detection mechanisms. In addition, smart home networks with multiple IoT devices are easily exploited by the attackers, and they are equipped to mine cryptocurrency on behalf of the attacker. However, no prior works investigated the impact of cryptojacking malware on IoT devices and compromised smart home networks. In this paper, we first propose an accurate and efficient IoT cryptojacking detection mechanism based on network traffic features, which can detect both in-browser and host-based cryptojacking. Then, we focus on the cryptojacking implementation problem on new device categories (e.g., IoT) and designed several novel experiment scenarios to assess our detection mechanism to cover the current attack surface of the attackers. Particularly, we tested our mechanism in various attack configurations and network settings. For this, we used a dataset of network traces consisting of $6.4M$ network packets and showed that our detection algorithm can obtain accuracy as high as 99% with only one-hour of training data. To the best of our knowledge, this work is the first study focusing on IoT cryptojacking and the first study analyzing various attacker behaviors and network settings in the area of cryptojacking detection.**

I. Introduction

Blockchain technology eliminates the central authority and ensures the immutability of on-chain transactions with a computational power-based consensus model. This consensus model is known as Proof of Work (PoW), and it is used by several blockchain networks such as Bitcoin [1], Ethereum [2], and Monero [3]. PoW consensus algorithms depend on the computational power of the hardware such as CPU, GPU, and dedicated chipsets designed to mine cryptocurrencies (e.g., ASIC miners). These components help miners to solve difficult hash-based puzzles. Solving the hash-based puzzle consumes a great amount of energy, and this energy cost is one of the main expenses in cryptocurrency mining operations. In this ecosystem, *cryptojacking* is an act of using victims' processing power without their knowledge and consent [4], [5]. There are two main approaches that attackers abuse the computational power of the victims for cryptojacking [4]. They either inject the script into the website or spread the mining program into the host machine. In both approaches, attackers exploit system vulnerabilities as well as traditional methods like social engineering techniques.

Cryptojacking attackers initially targeted personal computers by embedding the mining script inside popular websites [4]. While the cryptojacking attacks gain more popularity and target bigger attack domains [6]–[10], IoT devices became the new favorite toy of the attackers [6], [11], [12] because of their increasing usage in various different domains such as healthcare, industry, home, offices. IoT devices are generally resource-constrained, i.e., they are not profitable individually for an attacker. Therefore, the attackers utilize techniques like botnet attacks to take control of the IoT devices at scale and equip them to perform cryptocurrency mining on behalf of the attacker. The famous IoT botnet malware, Mirai [13], performed a massive Distributed Denial of Service (DDoS) attack in 2017, and some other attacks [12], [14], [15] followed the footsteps of the Mirai botnet. Moreover, the attacker(s) who performed Mirai botnet attack also used this network to mine Bitcoin and turn the botnet network into a giant cryptojacking mining pool. Just recently, another Mirai-inspired botnet, LIQUOR IoT botnet [15] started to mine Monero on its victims' IoT devices. While the IoT industry and capabilities of IoT devices continue evolving, it also gives attackers more space to widen their attacking surface.

There are several reasons why IoT networks are indeed lucrative targets compared to non-IoT networks (e.g., regular computers, servers, corporate computers). First of all,

[§]Both authors contributed equally.

Network and Distributed Systems Security (NDSS) Symposium 2022
24-28 April 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.24208
www.ndss-symposium.org

the diversity of the vendors, communication protocols, and hardware makes it very difficult to develop standard/unified defense solutions while greatly increasing the attack surface for the attackers. Second, they are not easily configurable due to their limited capabilities and resources. For example, while the PCs, servers, or corporate devices can be easily configured to avoid such attacks, the IoT devices constantly communicate with the cloud for real-time and remote access capabilities, which provides another attack surface to the attackers to spread and keep the botnet network. Last but not least, security flaws of IoT devices (e.g., default passwords, no authentication) due to their rapidly growing market and lack of security awareness makes them an ideal target for cryptojacking malware attackers. Here, in this paper, we focus on IoT devices, which can be targeted by both in-browser and host-based cryptojacking malware. Therefore, we aim to design a detection system that can detect both in-browser and host-based IoT cryptojacking malware.

The detection of IoT cryptojacking is challenging because most of the IoT devices do not allow to be programmed to collect the hardware-level features [16]–[19] or browser-specific features [16], [19]–[28]. However, the network traffic-based features can be collected in a unified interface on the router, i.e., the devices do not have to be programmed or modified at all. Therefore, in this paper, we used in-site network-based features to detect the IoT cryptojacking malware and propose an accurate, lightweight, and easy-to-implement cryptojacking detection system that can detect both kinds of cryptojacking attacks without interfering with the hardware or software of the devices inside the home network.

We performed an extensive set of experiments to design and evaluate the optimum IoT cryptojacking detection mechanism. We first performed experiments to find the best-ranked features, the most accurate classifier, and the optimum training size. Then, we evaluated the effectiveness of our IoT cryptojacking detection mechanism with 12 novel experiments designed to assess various attacker behaviors and network settings. For this, we implemented the cryptojacking malware on IoT devices, a laptop, and a server in a safe setup. We explained several practical issues we came across during the implementation of cryptojacking on IoT devices in Section V-C.

Contributions. We summarize the major contributions of this paper as follows:

- We propose an accurate and efficient cryptojacking detection algorithm targeting IoT networks. Since we use network traffic-based features, our algorithm is capable of detecting both in-browser and host-based cryptojacking malware and can detect the cryptojacking malware with 99% accuracy with one-hour of training data without any dependence on cloud or on-device features.
- To evaluate our algorithm, we designed several novel experimental scenarios. We assessed both various attack configurations (i.e., cryptojacking types, profit strategies, victim devices, and throttle values) and network settings (e.g., fully or partially compromised). To the best of our knowledge, this paper is the first to analyze various attack strategies and network settings in the area of cryptojacking detection.
- To overcome some of the practical issues during the implementation of cryptojacking malware on the IoT devices, we used novel techniques, which can be adapted by other studies in the future.
- In order to accelerate the research in this area, we released both the dataset and code.[1]

Summary of findings. In addition to our lightweight and highly accurate IoT detection mechanism, extensive experiments we performed to assess various attacker behaviors and network settings, which led to several interesting results worth nothing:

- We found that the highest malicious packet generation rate is 72% less than the least packet generation rate of the benign dataset given in Table III. This shows that the cryptojacking malware does not generate as many packets as daily web browsing and application data.
- We found that while in-browser cryptojacking malware uses evasion techniques such as limiting CPU and minimizing the network communication, host-based cryptojacking malware tries to take advantage of the victim's device at maximum computational power.
- We observed that an attacker targeting the server shows higher accuracy than other victim devices types (i.e., laptop and IoT), i.e., there is a higher chance that the cryptojacking attacker will be detected during an attack targeting the server type device.
- We found that the malicious scenario with stealthy profit strategy (i.e., 10% throttle) is less accurate than robust (i.e., 50% throttle) and aggressive (i.e., 100% throttle) attack scenarios. This means that the obfuscation methods of the attackers can still create differences during the detection phase.

Organization. The remainder of this paper is organized as follows: In Section II, we give background information for cryptocurrency mining, cryptojacking malware, and machine learning tools we used in the experiments. Section III defines the adversarial model and attack scenarios. Section IV presents our modeling to convert the network traffic data into a binary classification problem. Then, in Section V, we explain the details of our data creation process and perform initial data analysis on the raw dataset. Section VI reports the novel design scenarios to evaluate our detection mechanism and the results. Section VII discusses some of the challenges we came across during the implementation of the IoT cryptojacking malware and our novel techniques to overcome them. In Section VIII, we present the related work. Finally, Section IX concludes the paper.

[1] https://github.com/cslfiu/IoTCryptojacking

II. Background

A. Cryptocurrency Mining

Cryptocurrency mining is the process by which new cryptocurrencies enter circulation and is a critical component of the maintenance and continuity of the distributed blockchain ledger. The immutability of a blockchain network is provided by the consensus mechanism, which is cryptocurrency mining. Cryptocurrency mining is based on a puzzle based on the main features of cryptographic hash algorithms. These work-based consensus models are generally known as Proof of Work (PoW) consensus models.

Cryptocurrency mining is a laborious, costly process where one's reward depends on the luck factor. Work-based consensus mechanisms benefit from the diffusion feature of the hash algorithms to prevent miners from predicting hash values in a systematical pattern, and this feature also maintains the luck factor. However, due to the fact that miners are rewarded with cryptocurrencies for their work, it is an important source of income for many cryptocurrency investors. As the hardware investment increases, the difficulty, cost, and risk increase, while the chance of finding the block increases.

B. Cryptojacking Types

This section explains the details of different types of cryptojacking malware and their similarities and differences.

1) In-browser cryptojacking: The fast development of web technologies such as JavaScript (JS) programming language libraries and WebAssembly (Wasm) open standards allow web developers to interact with the computer components via several instructions in the browser. The attackers also use these technologies to implement in-browser cryptojacking malware. For example, Wasm provides the capability to run low-level instruction codes near-native speeds in browsers, and it is supported by all major browsers [29], [30].

In-browser mining surged after the service providers (e.g., Coinhive) started to distribute easy-to-use cryptojacking scripts. With those scripts, the attackers can easily copy and paste an HTML script to the source of the webpage via several code injection vulnerabilities [31]. This piece of code creates a communication pipeline for the mining process and starts the mining process. In-browser cryptojacking scripts also include an identification number of the script owner. With this ID, service providers monitor the overall traffic coming from the specific script owner and make the reward distribution. After the mining process started, the communication continues to receive the tasks and return the calculated results through the already established channel. The rewards are given to the account associated with the ID number periodically.

2) Host-based cryptojacking: Host-based cryptojacking malware aims to hide itself into the computer system of the victim's host and performs cryptocurrency mining. The main goal is placing the cryptojacking malware into the computational device of the victim as long as possible and keeping it profitable.

The attackers distribute and locate their host-based cryptojacking malware with third-party applications [32], [33], social engineering methods [34], or using several vulnerabilities into victims' host system [35]. Attackers also use IoT botnets [36] to perform cryptojacking attacks. IoT devices have limited capabilities in terms of computational power. However, the incentive of the attackers is similar to the botnets here, in which the combined computational power of a large number of IoT devices can be used to perform a meaningful amount of cryptocurrency mining.

C. Machine Learning Tools

In this subsection, we explain the machine learning algorithms and methods we used during our experiments.

1) Feature Extraction and Selection Tools: Feature extraction is a dataset size reduction operation where an initial raw dataset is reduced to a more manageable and usable form for processing. Feature extraction methods aim to combine features with different property-based functions, effectively reducing the size of data that classifiers need to process and still describing the original dataset without any loss. There are several open-source tools that calculate thousands of different features automatically. In this paper, we used *tsfresh* [37] automatic feature calculation tool.

Feature selection is the next step of the feature extraction process. After the feature extraction tools calculate the features, the tools sort them in terms of their significance level [38] (also known as P-value) and build relevance table [39]. With this method, we can easily eliminate the less significant features and improve our classification process.

2) Machine Learning Classifiers: Classification is the process where the algorithms categorize data into a given number of classes for the purpose of predicting the class of a given data feed. We used several different classification models (e.g., Logreg, KNN, SVM, RF) to train our models and receive the accuracy results in this paper.

There exist other cryptojacking detection and prevention methods such as *blacklisting* and *static analysis* methods like keyword detection. Blacklisting methods can be easily bypassed by the attackers by changing their domains. Even though most of the attackers obtain the source code for the cryptojacking from the same service provider, the malware can be easily embedded into their own websites. Moreover, the blacklists are generally not updated. On the other hand, the keyword-based detection systems cannot be used directly to detect the binary cryptojacking malware and can be easily bypassed using the obfuscation techniques. Additionally, IoT devices require a constant communication with the cloud and the cloud applications may use hosting services and dynamic IPs. Therefore, blocking the Internet or whitelisting/blacklisting the IP addresses regularly is not a feasible approach. However, a ML-based detection system would be more scalable and generalizable over time.

III. Adversary Model and Attack Scenarios

As mentioned before, IoT devices can be targeted by both in-browser and host-based cryptojacking malware. Attackers may also follow the path of different adversarial models and

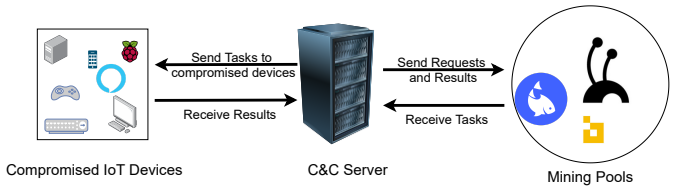

Fig. 1: Communication flow of C&C server-based cryptojacking from the compromised IoT devices to the servers.

attack cases. We evaluated 7 attack cases and made 12 discrete experiments to test the cryptojacking detection mechanism we proposed in this paper. In this section, we explain how IoT devices are targeted by cryptojacking malware and how we track these adversaries in our experiments.

A. Cryptojacking with service providers

There are several different active service providers and thousands of web pages hosting cryptojacking malware on the Internet [4]. Attackers generally take advantage of code injection vulnerabilities [12] of the webpages and web applications to inject their ready-to-use mining scripts provided by service providers.

In recent years, IoT devices gain new capabilities to provide better and more flexible frameworks to their users. These capabilities allow developers to integrate new technologies and run their code blocks via IoT frameworks. The attackers merge these framework capabilities with known vulnerabilities and abuse them to run their cryptojacking malware inside of these devices. We implemented WebOS IoT cryptojacking malware with LG's WebOS development framework [40] and develop a basic WebOS application that calls cryptojacking script when the user starts running the application.

To be able to create a stable and controlled cryptojacking environment, we prepared a website under a controlled server and hosted several different *active* cryptojacking scripts. However, while some of the service providers can not provide a stable mining framework, we chose Webmine [41] as our main service provider. We ran the script with combinations of different levels of computational hardware usage to observe the characteristic outcomes of those scripts.

B. Cryptojacking with Command and Control (C&C) servers

A C&C is a computer controlled by the attacker to send commands to compromised devices. Attackers generally host these servers in cloud-based platforms for security and identity secrecy reasons. In the cryptojacking domain, C&C servers are working as a subset of a mining pool to receive and distribute mining tasks from the mining pool to compromised devices. Figure 1 represents a basic configuration of the C&C servers connected to the mining pools. The first well-known incident related to IoT botnets and cryptocurrency mining [12], [36] happened in 2017 under the famous Mirai IoT botnet [4], [13], [42].

In this paper, we focused on the communication pipeline between the compromised device and the C&C server. To demonstrate the process and data communication in this setup, we created a C&C server that sent mining tasks between different time periods. This time frequency can be changed depending on the block frequency of the blockchain network. In this work, we focused on Monero [3] and sent a mining task package within every two minute frequency. We successfully implemented this scenario with LG WebOS [40] Smart TV and other platforms we used for testing purposes.

IV. IoT Cryptojacking Detection via Network Traffic

Network traffic classification and identification techniques have gained a lot of popularity in the last several years and it is a well-known technique to create user/device profiles on both server and local network sides [43], [44]. In this paper, we consider smart home network settings, where many IoT and non-IoT devices are connected to a router to be able to connect to the Internet. Each device can be identified via its MAC address. Therefore, we define devices in the network as $(MAC_0, MAC_1, ..., MAC_n)$ for given n devices in the network. We assume one or more devices in this network are compromised by the attacker to perform cryptocurrency mining on behalf of the attacker and our purpose is to detect the devices performing by monitoring their network traffic for a certain time duration. For this, we use machine learning algorithms, which are trained with malicious and benign data beforehand. Devices generate continuous network traffic, which needs to be converted into a data format where the machine learning algorithm can predict whether the device is performing or not. Before converting packets into the proper format, we filter each packet using the following filter:

$$(MAC_{src} == MAC_i)\, OR\, (MAC_{dst} == MAC_i) \quad (1)$$

for a given device with MAC address of MAC_i. Then, we extracted following metadata from each packet:

$$Pkt_i = [MAC_i, timestamp, packet\ length] \quad (2)$$

At the end of this process, we obtain a series of packet lengths arrived at a given time for every device. Finally, we use a burst of 10 packets to calculate the features and we use these features to train/test the machine learning algorithm.

V. Dataset Collection

The data we focused on in the paper is the network communication data between the IoT devices and cryptojacking service providers. In this section, we explain the main dataset collection and creation process by focusing on the topology, tools, methodology, and other implementation details we used for the IoT environment environment.

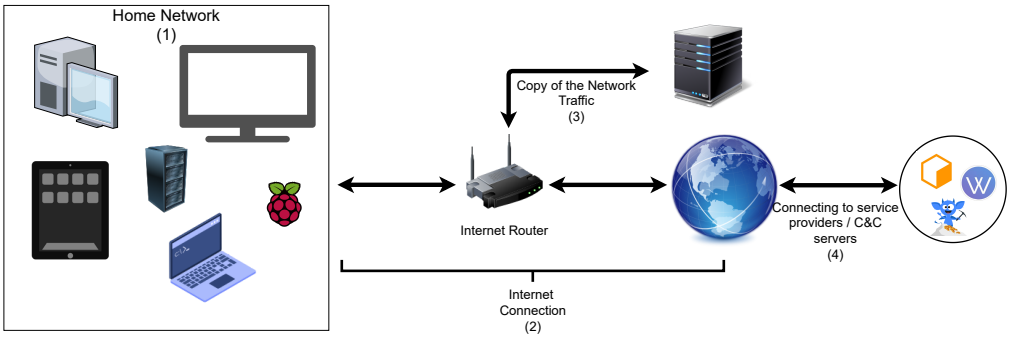

Fig. 2: General flow of network data collection process.

A. Topology

Figure 2 demonstrates our reference topology. We created a regular smart home network with several smart home devices, IoT devices, and personal computers (1). Data collection was performed under regular home-networking settings and all the devices performed unauthorized mining under a controlled environment. In our setup, all the devices in the home network are connected to the Internet via a single Internet router (2) similar to most home settings. There is also another computer in the network dedicated to collecting all the Internet traffic with port mirroring [45] and ARP rerouting/poisoning [46] techniques (3). Finally, compromised devices in the network connect to the cryptojacking service providers or C&C servers of mining malware (4) to receive the tasks and return the calculated results. In this topology, our purpose is to be able to detect the compromised smart home devices that are performing unauthorized cryptomining inside the network.

B. Devices

We performed our experiments on four different devices representing varying computational power. Particularly, we performed the experiments on Raspberry Pi, Laptop, Tower Server, and LG Smart TV. Raspberry Pi and LG Smart TV represent the IoT devices in a real-life network while Laptop represents a regular device and Tower Server represents a computationally powerful device. Table I shows the devices we used during the experiments and their specifications.

Device	Representation	Hardware	Operating System
Raspberry Pi	IoT device	Cortex-A72 64-bit SoC 4GB RAM	Raspberry OS
LG Smart TV	IoT device	LG Quad Core Processor	WebOS 2.0
Laptop	Regular Device	Intel Core i7 9th Generation CPU 16 GB DDR4 RAM	Ubuntu 18.04 LTS
Tower Server	Powerful Device	Intel® Xeon® Gold 6314U Processor 192 GB DDR4 RAM	Ubuntu 20.04
Router	Internet Routing	Atheros QCA9563 Processor	OpenWRT V.19.07.1

TABLE I: Device list used in the experiments.

Moreover, we used TP-link Archer C7 V5 as our router in the given topology in Figure 2. We also used Ettercap [47] to manipulate the ARP protocol and forward the network traffic to the data collection computer's IP address. With this network configuration, we were able to collect all networking data with the Wireshark packet collector and analyzer [48].

C. Implementation Methodology

The implementation of in-browser and host-based cryptojacking differ in several ways. In the next subsections, we explain the details of their implementations.

1) Implementing In-browser Cryptojacking: To be able to implement in-browser cryptojacking under a safe environment, we launched a basic WordPress [49] webpage which contains several different cryptojacking malware. We placed different HTML-based cryptojacking malware samples inside the source code of different pages of the test website for our purposes. After creating our experiment setup, we connected these pages with our test device and collected network traffic data for at least 12 hours for every use case scenario as explained in Section III. We used the scripts distributed by Webmine.io [41] and WebminePool [50] service providers for the in-browser cryptocurrency mining.

2) Implementing Host-based Cryptojacking: Implementing host-based cryptojacking on Raspberry Pi, Laptop, and Tower Server is straightforward. We downloaded the cryptocurrency mining binary MinerGate V1.7 [51] and run it on our test device with our configurations. However, the implementation of host-based cryptojacking on the LG Smart TV was more challenging because the malware binary had to be placed on the victim's device and the victim's device had to allow to run the executable sample. We used the LG WebOS developer framework [40] to develop a basic application that runs cryptojacking malware as long as the application was running. In our scenario, we assumed that the malicious application could be an IP TV or streaming application. As a C&C server, we used a basic cloud server (i.e., 1 GB RAM, 1 Core CPU, and Ubuntu Server 18.4). After we created the malicious application that runs inside the WebOS-supported Smart TVs and the C&C server setup, we implemented two different models for the actual mining process as follows;

- **Without connecting mining pool:** We made the first implementation with basic RandomX PoW algorithm [3], [52]. When the application is activated, it sends a connection request to the C&C server, after that, the C&C server sends the mining tasks to the malicious application. The mining task contains three variables, the hash value, nonce value range, and the difficulty target. The RandomX implementation inside the malicious application

starts mining inside the Smart TV until new command comes from the C&C server or finishes the nonce value range or finds the hash and nonce value that meet with the difficulty target.

- **Connecting mining pool via API:** The difference between this implementation and the previous one is that the C&C server does not create mining tasks by itself, it receives them from the mining pool via its API framework and sends them over to the malicious application.

The only difference between the two methods is the entity creating the mining tasks (C&C server vs. mining pool). The messages exchanged between the C&C server and malicious application are the same. Therefore, for our dataset, we used Ant mining pool [53] and collected the data flow between the malicious application and the C&C server. In terms of the attackers, both methods have various advantages/disadvantages. Attackers who run botnet platforms (i.e., botnet admins) with anonymity priority may want to create a self-mining environment. However, running a self-mining environment requires having a synchronized full node and a lot of extra labor to maintain. On the other hand, using the mining pool API is a lot more convenient and allows important extra flexibility to the attackers while providing degraded anonymity. Please note that both of the implementations we presented are just created for proof-of-concept purposes.

D. Labeling

While Wireshark was collecting all networking data, we ran all the attack scenarios we covered in Section III to collect the networking data. The network traffic generated by a device performing mining is labeled as malicious, while the dataset that is collected by a device that is not performing cryptocurrency mining is marked as benign.

E. Initial Data Analysis

We designed different scenarios to collect malicious data with our controlled environment setup to assess the various configurations that an attacker may use and network settings that are possible in real-life smart home environments. More details about the configurations and our results are given in Section VI. On the other hand, for the first set of experiments, we downloaded the benign dataset from a public repository [54] and for the second set of experiments, we collected our own benign dataset for the same set of devices that we used for malicious data collection. The full details of the dataset are presented in Table II, III, and XI.

1) Benign vs. Malicious: The main goal of this paper is to be able to differentiate the malicious and benign networking data from each other. For this purpose, we performed some initial data analysis on the cryptojacking networking data and list the outcomes as follows:

- **Packets Per Second (PPS) rate** is an important statistic to differentiate between the malicious and benign data. As we can observe from Table II, the highest PPS rate is produced by the most powerful device we used while it was running binary cryptojacking malware. However, the highest malicious PPS rate is still 72% less than the least PPS rate of the Benign Dataset given in Table III. This shows that the cryptojacking malware does not generate as many packets as daily web browsing and application data. This is an important challenge for both the data collection and analysis phases. We discussed this challenge more in Section VII.
- **Average Packet Size (APS) rate** is the average size of all inbound and outbound network packets. It is a meaningful and relevant feature for purpose of the data discrimination process. APS rate creates almost the same pattern as the PPS rate. The highest malicious PPS rate is created by the Raspberry Pi while performing in-browser mining with webmine.io [41] service provider but the highest APS rate of the malicious data is still 35% less than the least APS rate of the benign data.

To sum up, network communication of cryptojacking malware and daily benign user shows very different characteristic features. In the rest of this paper, we use this evaluated knowledge.

2) Host-based vs. In-browser Cryptojacking: The attackers mainly perform two different cryptojacking attacks to target different domains, in-browser and host-based cryptojacking attacks [4]. To be able to see the different patterns generated by different devices under different attack scenarios, we performed in-browser and binary cryptojacking on all devices we used in this paper and summarize the results in Table II. We can summarize our observations as follows:

- **In-browser mining** always tends to generate a very small amount of PPS rate and APS rate on all devices.
- **For different service providers** of in-browser mining, there is no significant difference between the two service providers we used (i.e., Wembine.io and WebminePool).
- **Binary mining** samples do not seem to use the obfuscation features used by the in-browser mining applications. They do not have any intonation to minimize their communication and keep themselves as stealth as possible.
- **For both in-browser and binary** mining patterns, we can observe that in-browser mining always creates a very little amount of network traffic. There is no significant correlation between the hardware power, PPS rate, and APS rate. However, binary mining shows a completely different pattern where the APS and PPS rates are directly correlated with the power of the device.

3) Raspberry vs. Laptop vs. Server: Finally, we made the device-specific analysis to be able to see if there is any relevant feature that may allow us to differentiate the devices and understand which specific device category is infected by cryptojacking malware. We summarize our observations as follows:

- **All devices** give almost the same PPS and APS results for in-browser mining applications. We can deduce from the dataset outcomes, the service providers firstly receive the capabilities of the victims' host system and send mining tasks in terms of the power of the victims' host device.

Dataset Name	Cryptojacking Type	Device	Software	Attacker	Currency	Total time (Minutes)	Packet Count	Packets Per Second (PPS)	Average Packet Size (Bytes) (APS)
Raspberry_Webmine.io_Robust	In-browser	Raspberry Pi 4	Webmine.io	Robust	Monero	52	3621	1.2	**479**
Raspberry_Webmine.io_Aggressive	In-browser	Raspberry Pi 4	Webmine.io	Aggressive	Monero	735	14156	0.3	**163**
Raspberry_WebminePool_Stealthy	In-browser	Raspberry Pi 4	WebminePool	Stealthy	Monero	521	10285	0.3	146
Raspberry_WebminePool_Robust	In-browser	Raspberry Pi 4	WebminePool	Robust	Monero	527	7708	0.20	141
Raspberry_WebminePool_Aggressive	In-browser	Raspberry Pi 4	WebminePool	Aggressive	Monero	1080	24476	0.40	145
Server_WebminePool_Robust	In-browser	Server	WebminePool	Robust	Monero	382	18460	0.8	498
Server_WebminePool_Aggressive	In-browser	Server	WebminePool	Aggressive	Monero	60	3106	0.9	297
Desktop_WebminePool_Aggressive	In-browser	Desktop	WebminePool	Aggressive	Monero	726	234892	5.4	3128
Raspberry_Binary	Host-based	Raspberry Pi 4	MinerGate	Aggressive	Monero	983	22111	0.4	95
Server_Binary	Host-based	Server	MinerGate	Aggressive	Monero	1024	1213354	**19.7**	154
WebOS	Host-based	LG Smart TV	AntMiningPool	Aggressive	Monero	61	43173	**11.80**	242
Total				**6145**	**1558831**				

TABLE II: Malicious dataset sample sizes.

Dataset Name	Domain	Total time (Minutes)	Packet Count	Packets Per Second (PPS)	Average Packet Size (Bytes) (APS)
Bulk Data	Internet Data	18	2204727	2636.50	1114.5
Web Multiple	Internet Data	14.56	95388	91.78	567.25
Interactive	Internet Data	20.33	26144	355.97	249
Video	Internet Data	9.55	140009	243.33	956.3333333
Web Single	Internet Data	12.08	51381	71.46	638
Total		**74.52**	**2517649**		

TABLE III: Benign dataset sample sizes.

- **For all devices** performing binary mining, we observe that the binary cryptojacking malware is correlated to the power of the victims' host system and both the PPS and APS rates are directly affected by the power of victims' host system as well.

To sum up, we found that while in-browser cryptojacking malware is trying to keep itself under a low profile and prevent high data density communication, host-based cryptojacking malware is not behaving in this way and generating huge network traffic. This is because of the method used by host-based cryptojacking malware, in which they are generally merged with other computationally heavy applications [4]. Therefore, the attackers are not worried about creating highly visible network communication.

VI. Evaluation

In this section, we designed four sets of experiments to design and evaluate IoT cryptojacking detection mechanism that is accurate, efficient, and works in varying configurations and network settings.

- First, we performed a set of experiments to design the optimum IoT detection mechanism with a highly accurate prediction rate and minimum training size and time.
- Second, we performed experiments to assess the proposed detection mechanism we designed in the first part for different configurations such as different devices and throttle values.
- Third, we performed a set of experiments to assess the proposed mechanism in various smart home network settings.
- Fourth, we performed a set of experiments to assess the sensitivity of the proposed classifier.

We explain the details of these experiments and our results in the following subsections.

A. Designing an IoT Cryptojacking Detection Mechanism

After the dataset collection and labeling process, we created an overall dataset that contains the combination of all the malicious datasets and a benign dataset with an equivalent number of packets. Table IV presents the dataset sizes and the total feature extraction and classification time of the overall dataset.

Dataset Name	Dataset size	Total Classification Time
Malicious	1557230	8.5 Hours
Benign	1556899	

TABLE IV: The Overall Dataset Sample Sizes and total Feature Extraction & Classification Time.

In this sub-section, our goal is to design an IoT detection mechanism based on network traffic features and utilizing Machine Learning (ML) classifier for the classification. For this purpose, we performed the following steps:

- We use Feature Extraction to create feature vectors from the raw dataset.
- We select the best features and remove irrelevant features via a Feature Selection algorithm.
- We train and test several ML classifiers and decide which one performs best.
- We test the best algorithm with varying training sizes to optimize the training data and time as well as calculate the prediction time to assess the algorithm's feasibility in a real-world application.

We explain the details of these experiments and their results in the following subsections.

1) Feature Extraction: We used tsfresh [37] to extract features from our dataset. The library tsfresh is a python package that automatically calculating statistical features from time-series data. In our case, we used ten packets for each feature vector and it calculates 788 different statistical features

from its wide selection of statistical features [55]. The features we used to train our overall dataset and other experiments are as follows:

- **Timestamp** is an important data extracted from every packet, and we used it to sort the given packets. While the mining process is running, data communication pattern changes depending on several variables. After we calculated the delta mean values of malicious and benign traffic datasets, we observed an 11.4% difference between the mean values. This significant time difference leads us to use the timestamps of the network packages as a feature. We also noted that the devices mostly transmit stay-alive packets when no result is returned while they return application data packets with payload. And, while stay-alive packets are mostly transmitted periodically, the interarrival time between the packets decreases significantly while transferring application data packets.
- **Packet length** is another extracted data from every packet. All the miners must return some positive or negative result to the pool network. If the miner is directly connected to the mining network, they need to use several special protocols such as the stratum mining communication protocol used by Ethereum.

2) Feature Selection: Feature selection is the process of selecting a subset of relevant features for our models. The relevancy scores of all features in our dataset are not the same. To be able to optimize our extracted features and use only the most relevant features, we calculated the P-value. P-value [56] is a statistical model that calculates the probability of finding an observation under the assumption that a particular hypothesis is true. A smaller P-value (less than 0.05) is considered statistically significant. We found 290 statistically significant features for our datasets and train our models with those features. We repeated the same process for the rest of the experiments.

3) Classifier Selection: We implemented four Machine Learning classifiers to test the accuracy of the features described in the previous subsection. During the implementation of these classifiers, we used 75% of the data to train and 25% to test the classifier. We used the default parameters of scikit-learn [57] for the first three set of experiments while we tested the non-default parameters in Section VI-D3. We used 5-fold Cross Validation (CV) for assessing the effectiveness of the classifier and used Accuracy, Precision, Recall, F1 Score, and Test_roc as our metrics for all experiments.

Classifier	Accuracy	Precision	Recall	F1 Score	Test ROC
Logreg	0.97	0.97	0.97	0.97	0.988
KNN	0.98	0.98	0.98	0.98	0.99
SVM	0.99	0.98	0.98	0.98	0.99
GNB	0.96	0.96	0.96	0.96	0.97

In this table, we used weighted average values calculated by scikit learn libraries.

TABLE V: The Overall Dataset Classification Results.

The results of our overall dataset experiment are presented in Table V. Our results showed that SVM performs the best among all the classifiers in terms of all metrics, aligning with many other detection studies in the literature [4]. SVM is a useful and well-designed classifier for supervised machine learning and it is very effective when there is a clear margin of separation. In addition, the SVM classifier is very stable, and small changes in the dataset do not cause important changes in results. Therefore, we decided to use SVM classifiers for our further analysis and implementation of other use case scenarios in the rest of the paper.

Dataset		Dataset Sample Sizes			
		12 hours	**6 Hours**	**3 Hours**	**1 Hour**
Server	Malicious	838627	419313	209656	69885
	Benign	837701	418850	209425	69808
Desktop	Malicious	234272	117136	58568	19522
	Benign	234448	117224	58612	19537
Raspberry	Malicious	7829	3914	1957	978
	Benign	8265	4132	2066	1033

TABLE VI: Dataset sizes of timing experiments.

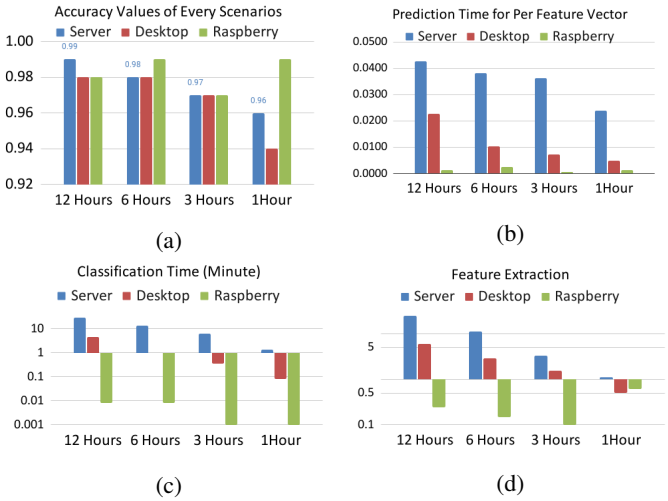

Fig. 3: **(a)** The accuracy, **(b)** prediction times for every one feature vector during classification (in seconds), **(c)** classification time used for training (in minutes), and **(d)** the time used for feature extraction (in minutes) for each training size (i.e., 1, 3, 6, and 12 hours).

4) Training Size and Timing Results: In this section, we performed experiments with varying training sizes. With this experiment, we analyzed the effects of the dataset collection time on classification accuracy and overall classification time. To obtain a reference result, we firstly fit the times of representative datasets to 12 hours by decreasing the size of the original dataset and fit them into 12 hours, 6 hours, 3 hours, and finally 1 hour and repeat the classification to measure accuracy and time-based values for each training size. Figure 3 summarizes 4 different results we cover under this section.

- **Accuracy** is the first metric we checked after we rerun the classification for every dataset. As can be seen from the results in Figure 3a, the accuracy did not change dramatically, and even when we only used less than 10% of the original dataset such that we did not receive any

result below 94%. In addition to this, our model achieved to detect in-browser cryptojacking with 99% accuracy with only one-hour long data collection. It shows that the model we used is not extremely dependent on the dataset size, and it can give accurate results even with a shorter data collection duration.

- **Prediction time for per feature vector** represents how much time we need to predict the class per feature vector. This is a realistic metric we calculated to see how long it would take to predict the result of the collected dataset on a regular machine. After we evaluate our experiments, we saw that the time needed to train per feature vector is related to the size of the dataset. For bigger datasets, it takes more time to evaluate per data. However, after the feature extraction process, we receive successful optimization results as low as 100-150 ms for each vector.
- **Feature extraction and classification time** represents the needed time for calculating features and classify these features for each dataset. As can be seen from Figure 3c and 3d, the total time needed for the feature extraction and classification is directly correlated to the dataset size. In addition to this, with the very low feature extraction and classification time results, we still managed to get near-perfect results.

Overall, the results show that we achieved to implement a successful detection system without causing a lot of overhead on the devices or inside the network. We can also conclude that we can use slightly smaller datasets to train our model without sacrificing our dataset's accuracy level and trust factor.

B. Evaluation With Different Adversarial Behaviours

In this sub-section, our goal is to assess the IoT cryptojacking detection mechanism we designed in Section VI-A with various attack configurations. An attacker can target different victim devices, pursue different profit strategies or have a choice of cryptojacking type of either in-browser or host-based. We have extensively evaluated our mechanism by performing a comprehensive set of experiments to test these three configurations. All the scenarios and experiments are implemented using the same feature extraction and selection process described in Section VI-A. This implementation methodology allows us to observe the results of how effective to use one feature set for different use case scenarios. We created a balanced dataset for three scenarios to minimize the effect of the imbalanced dataset issues. Table VII presents the dataset sizes and sources we used to implement these three scenarios. We used the SVM classifier for the model training process. In the rest of this section, we summarized the detailed experiment results for each scenario.

1) Scenario 1 - Server vs. Desktop vs. IoT: In this scenario, we set up our environment with different kinds of devices. Our goal is to see if there are any differences between the detection accuracy of the cryptojacking malware running on each device. For this purpose, we created a balanced dataset for each device.

	Attack Case	Benign	Malicious
Scenario 1	Server	1212596	1217322
	Desktop	234448	234272
	IoT	64064	65030
Scenario 2	Aggressive	1519162	1520681
	Robust	27915	26669
	Stealthy	9877	9880
Scenario 3	In-Browser	305986	305874
	Host-Based	1250241	1251356

TABLE VII: Dataset sizes used in the experiments.

2) Scenario 2 - Profit Strategies: Throttle adjustment is one of the major obfuscation methods used by the attackers [4] and almost all of the active and inactive service providers provide this option to their clients. With the throttle adjustment feature, attackers can set the total hardware usage on victims' hots devices. It makes this feature an important use case for our detection mechanism. While increasing the throttle value increases the profit, it also increases the likelihood of being detected by the system. Our goal with this experiment is to see if changing the throttle value has any impact on the detection accuracy.

- **Aggressive** cryptojacking malware focuses on making a maximum profit in minimum time. They are generally seen on the websites visited for a short time (less than 2 minutes) such as e-commerce, dictionary, and legal/illegal data downloading websites. The aggressive cryptojacking malware parallels the mining task to use all the remaining CPU power; therefore, it deteriorates the user experience dramatically. Thus, the malware detection algorithms or even the user can easily notice and detect the cryptojacking malware. We set the throttle value to 100% to observe such an attacker behavior.
- **Robust** cryptojacking malware employs multiple scripts to keep working even if the primary service provider fails. Service provider failure can occur not only on the server-side, but also on the victim side. For example, some extensions may block the connection to a service provider. In this case, the script can continue mining with other service providers via this strategy. We set the throttle value to 50% to observe such an attacker behavior as it only runs one of the scripts at a time.
- **Stealthy** cryptojacking malware aims not to be noticed by the user and by threshold-based detection algorithms deployed on the user side. It utilizes CPU limiting for long-term profit and is widely seen on illegal media/content websites (e.g., illegal movie/series streamers, reading content pages, forums), where the users tend to spend a relatively longer time. It acts as an active content provider by using the same amount of CPU power as online flash games and media streamers. Therefore, it is hard to detect. We set the throttle value to 10% to observe such an attacker behavior.

3) Scenario 3 - In-Browser vs Binary: Host-based and in-browser are the two main cryptojacking type. Our goal, in this experiment, is to see if our detection system can success-

fully classify the in-browser and the host-based cryptojacking malware.

4) Results: Table VIII presents the accuracy results of all three different attacker cases. In all three scenarios, we observed some differences between the different configurations.

	Attack Case	Accuracy	Precision	Recall	F1 Score	Test ROC
Scenario 1	Server	0.99	0.99	0.99	0.99	0.99
	Desktop	0.98	0.98	0.98	0.98	0.99
	IoT	0.93	0.93	0.93	0.93	0.96
Scenario 2	Aggressive	0.98	0.98	0.98	0.98	0.99
	Robust	0.87	0.87	0.87	0.87	0.94
	Stealthy	0.91	0.92	0.91	0.91	0.98
Scenario 3	In-Browser	0.95	0.95	0.95	0.95	0.98
	Host-Based	0.99	0.99	0.99	0.99	0.99

In this table, we used weighted average values calculated by scikit-learn libraries.

TABLE VIII: Classification results of all scenarios.

In Scenario 1, we successfully received an almost perfect score from all three experiments we made. However, the server shows a higher accuracy among all victim device types, i.e., there is a higher chance that the cryptojacking attacker will be detected during an attack targeting the server type device. In Scenario 2, the malicious scenario with stealthy profit strategy (i.e., 10% throttle) and robust profit strategy (i.e., 50% throttle) are both less accurate than aggressive (i.e., 100% throttle) attack scenario. As we mentioned in Section III, attackers use these obfuscation methods to keep their miners safe from the detection methods. While 87% or 91% accuracy values can still be considered significantly high, it also means that obfuscation methods of attackers can still create differences during the detection phase. Finally, in Scenario 3, we can see the effect of the combination of obfuscation methods on in-browser cryptojacking detection samples. The general results for in-browser cryptojacking malware are just one step behind the host-based cryptojacking samples. While cryptojacking malware has the ability to infect different devices, the proposed malware detection system needs to be able to detect the ongoing cryptojacking process without any device dependency. We saw that our extracted features could achieve near-perfect scores without any device dependency from the results of three scenarios and eight discrete experiments.

C. Adversarial Models of Compromised Device Numbers in Smart Home Network

In this section, we investigate different adversarial models implemented inside a simulated network that can be seen in smart home environments. We implemented four different scenarios and presented their results in the rest of this section.

	Test Case	Malicious Dataset Size	Benign Dataset Size
Scenario 4	Fully compromised (All)	1557230	1556899
Scenario 5	Partially compromised (IoT + Laptop)	247143	246001
Scenario 6	Single compromised (IoT)	12781	13746
Scenario 7	IoT compromised (IoT + IoT)	47978	47801

TABLE IX: Dataset sizes and classification times of adversarial model analysis scenarios.

1) Scenario 4 - Fully compromised: In this scenario, we assume attacker(s) exploit all devices in the smart home environment. This scenario could apply to several network-based attacks. For this experiment, we used our overall dataset (i.e., all malicious data we collected). We give the dataset sizes in Table IX.

2) Scenario 5 - Partially compromised: Scenario 5 presents two different devices from different categories exploited by the attacker(s) with different cryptojacking attacks. While the IoT device was exploited by host-based cryptojacking and performed binary mining operations, the laptop device was compromised with an in-browser cryptojacking attack. In this scenario, most probably, one needs to consider two discrete attacks performed by different malicious entities. Still, in this scenario, both devices are using the same gateway (e.g., router, ADSL modem, Ethernet port) for Internet communication.

3) Scenario 6 - Single compromised: While attackers use specific vulnerabilities to inject their malware, there can be only one or very few devices that may be exploited by that specific vulnerabilities. When only one device in the network is compromised by the attackers, it makes it harder to detect the malicious device. Our goal, in this scenario, is to test the attack case where only one IoT device is compromised.

4) Scenario 7 - IoT compromised: Our last scenario is inspired by Scenario 3. In this scenario, we discussed what if two IoT devices from different domains were exploited by two different kinds of cryptojacking malware. To be able to simulate this environment, we used an LG WebOS smart TV exploited by a malicious application hosting the host-based cryptojacking malware and a Raspberry Pi that is exploited by the malicious webpage to perform in-browser mining. We received a near-perfect score for our last experiment as well.

	Test Case	Accuracy	Precision	Recall	F1-Score	Test ROC
Scenario 4	Fully compromised (Overall)	0.98	0.98	0.98	0.98	0.99
Scenario 5	Partially compromised (IoT + Laptop)	0.98	0.98	0.98	0.98	0.99
Scenario 6	Single compromised (IoT)	0.94	0.94	0.94	0.94	0.95
Scenario 7	IoT compromised (IoT + IoT)	0.92	0.92	0.92	0.92	0.96

TABLE X: Results of network settings analysis scenarios.

5) Results: In this section, we designed and implemented several scenarios to track a real home environment. We presented the results of these four scenarios in Table X. In Scenario 4, we assumed all of the devices are compromised. In Scenario 5, we used two different types of compromised devices that were exploited with different types of cryptojacking malware. In Scenario 6, only one device inside the home network performing a very limited amount of mining was compromised. Finally, in Scenario 7, we tried the same scenario with two different IoT devices (Raspberry Pi and WebOS Smart TV). The results of these last two scenarios have importance because these two scenarios are reflecting most of the Mirai [13] and other known IoT botnet [58] attack scenarios. Our result shows that the fully compromised scenario is the one most likely to be detected by our detection mechanism while it is harder to detect when only IoT devices

Dataset Name	Device	Activity	Total time (Minutes)	Packet Count	Packets Per Second (PPS)	Average Package Size (Bytes) (APS)
Laptop_idle_benign	Laptop	Idle	10.24	113602	184.9	929
Laptop_interactive_benign	Laptop	Interactive	22.1	81681	61.6	668
Laptop_webbrowsing_benign	Laptop	Web Browsing	11.43	99235	144.7	764
Laptop_download_benign	Laptop	Download	4.19	442866	1761.6	925
Laptop_video_benign	Laptop	Video	32.45	29010	14.9	1109
Raspberry_idle_benign	Raspberry	Idle	30.25	73	0	113
Raspberry_interactive_benign	Raspberry	Interactive	17.27	104241	100.6	764
Raspberry_webbrowsing_benign	Raspberry	Web Browsing	23.22	123298	88.5	946
Raspberry_download_benign	Raspberry	Download	4.11	276808	1122.5	1267
Raspberry_video_benign	Raspberry	Video	31.26	57205	30.5	1177
Server_idle_benign	Server	Idle	20.21	13459	11.1	142
Server_interactive_benign	Server	Interactive	18.01	123728	114.5	1143
Server_webbrowsing_benign	Server	Web Browsing	14.37	43713	50.7	1233
Server_download_benign	Server	Download	4.15	564831	2268.4	3438
Server_video_benign	Server	Video	14.18	109487	128.7	1069
WebOS_video_benign	WebOS	Livestream and Video	4.07	177704	727.7	930
Total			**261.51**	**2360886**		

TABLE XI: Our own benign dataset sample sizes.

are compromised, but we also see that all of the scenarios show a significantly high accuracy ($> 92\%$). This implies that the detection model and feature set we implemented can successfully detect various home environment attack scenarios. Overall, the combination of our selected classifier and feature set successfully detect the cryptojacking malware with high accuracy without being affected by any of the known attempts and obfuscation that may have been used by the attackers.

D. Classifier Sensitivity Evaluation

In this section, we performed more experiments to test the sensitivity of the classifier. For this purpose, we collected our own benign dataset using the devices given in Table I. Table XI shows the statistics of our benign dataset. Similar to the benign dataset we downloaded from a public repository [54], the PPS and APS rate of the benign dataset is much larger than the malicious dataset, which supports our finding about the high network traffic rate of benign activities compared to the cryptocurrency mining. On the other hand, in order to verify our results in the previous section with our own dataset, we repeated the same scenarios (Scenarios 1-7) and presented the dataset size and results in Table XVI and XVII in Appendix. Moreover, to test the sensitivity of our classifier, we design three more experiments: 1) Imbalanced Dataset, 2) Transferability, and 3) Non-default Parameters experiments. In the rest of the section, we explain the details of these experiments.

1) Scenario 8 - Imbalanced Dataset: We created the following datasets to test the imbalanced dataset scenarios:

- *Timely Balanced:* In this dataset, we used the four minutes data from all of the devices for both benign and malicious samples.
- *Timely Balanced with Oversampling:* In this dataset, we used the same dataset from the previous scenario (i.e., timely balanced) and used oversampling technique to handle the imbalanced dataset.
- *Same device:* In this case, we used the network traffic captured from the same device for both benign and malicious cases. Particularly, we tested the following cases:
  - *Server vs. Server*
  - *Laptop vs. Laptop*
  - *Raspberry vs. Raspberry*
  - *WebOS vs. WebOS*

All these cases resulted with various degree of imbalanced datasets. Table XII shows the exact number of benign and malicious dataset sizes for each cases. As can be seen from the table, timely balanced scenario resulted with an extremely imbalanced dataset with around $\sim 10k$ benign data and $\sim 1.5M$ malicious samples. We also used the oversampling technique, which is a commonly used technique to handle the imbalanced dataset. Moreover, we created imbalanced datasets using the data collected from the same device for benign and malicious samples.

	Attack Case		Benign	Malicious
Scenario 8	Timely Balanced		1634689	9741
	Timely Balanced with Oversampling		1634689	1634689
	Same Device	Server vs. Server	290397	1217322
		Laptop vs. Laptop	766394	234272
		Raspberry vs. Raspberry	561625	64064
		WebOS vs. WebOS	177704	41572

TABLE XII: Dataset sizes for the imbalanced dataset experiments.

Table XIII shows the performance of our detection system with different scenarios with imbalanced datasets. Our trained model was able to detect and correctly classify all imbalanced scenarios, achieving 98% overall accuracy.

	Attack Case		Accuracy	Precision	Recall	F1-Score	Test ROC
Scenario 8	Timely Balanced		0.99	0.99	0.99	0.99	0.96
	Timely Balanced with Oversampling		0.98	0.98	0.98	0.98	0.98
	Same Device	Server vs. Server	0.98	0.98	0.98	0.98	0.99
		Laptop vs. Laptop	0.99	0.99	0.99	0.99	0.99
		Raspberry vs. Raspberry	0.97	0.97	0.97	0.97	0.96
		WebOS vs. WebOS	0.97	0.97	0.97	0.97	0.99

TABLE XIII: Imbalanced dataset experiment results.

2) Scenario 9 - Classifier Transferability: In this set of experiments, we aim to measure the transferability of our

model and see how our detection system works against new/undiscovered attack surfaces. For this purpose, we trained and tested our model with different cryptojacking malware as well as different devices to test if our model is device- and malware-agnostic. The results of these scenarios are important to show if a pre-trained detection algorithm can be used for other cryptojacking malware or devices.

	Attack Case	Training Dataset		Testing Dataset	
		Name	Size	Name	Size
Scenario 9	Service Provider-1	Webmine	12871	WebminePool	19643
	Service Provider-2	WebminePool	19643	Webmine	12871
	Binary-1	IoT	11745	WebOS	41572
	Binary - In-Browser - 1	IoT	11745	IoT(Aggressive)	12871
	Binary - In-Browser - 2	IoT	11745	IoT(Robust)	3519
	Binary - In-Browser - 3	IoT	11745	IoT(Stealthy)	9880
	In-Browser - 1	IoT (Aggressive)	12871	IoT (Stealthy)	9880
	In-Browser - 2	IoT (Aggressive)	12871	IoT (Robust)	3519

TABLE XIV: Dataset names and sizes used in the transferability experiments.

	Attack Case	Accuracy	Precision	Recall	F1-Score	Test ROC
Scenario 9	Service Provider-1	0.87	0.92	0.87	0.88	0.93
	Service Provider-2	0.69	0.92	0.69	0.75	0.97
	Binary-1	0.87	0.84	0.87	0.81	0.99
	Binary - In-Browser - 1	0.90	0.90	0.90	0.89	0.99
	Binary - In-Browser - 2	0.99	0.99	0.99	0.99	0.99
	Binary - In-Browser - 3	0.99	0.99	0.99	0.99	0.99
	In-Browser - 1	0.99	0.99	0.99	0.99	0.99
	In-Browser - 2	0.97	0.96	0.97	0.96	0.99

TABLE XV: Results of the transferability experiments.

As can be seen from Table XV, our proposed detection system can detect cryptojacking malware without any platform and service provider dependency. These results verify that our machine learning-based detection system can provide an effective protection with only using network traffic as our features and can cover both the existing devices and new devices that will be released by the manufacturers even if the attackers are going to use new attack surfaces in the long run.

3) Scenario 10 - Non-default Parameters: In this section, we tested our classifier (i.e., SVM) with non-default parameters. We basically adjusted three parameters: Kernels, Regularisation parameter (C), Gamma. Kernel is the main function to transform low dimensional data into a higher-dimensional data, while the regularisation parameter is a penalty parameter used to tune between the decision boundary and classification error. On the other hand, when gamma parameter is high, nearby points will have a higher influence. We performed experiments with various values of kernels, regularisation parameter, and gamma. For these experiments, we selected the robust cryptojacking dataset captured from an IoT device. We present the results in Table XVIII in Appendix. Our results show that different variables can change the results of the SVM classifier dramatically. With the default parameters, the classifier trained the model with 87% average score. However, after we change the parameters and calculated the all possible 15 combinations, we received variety of training scores between 0.52-0.89. In addition to that, some of the parameters caused over-fitting on dataset classification. However, we note that our cryptojacking detection mechanism is highly configurable and can be easily customized depending on the needs, conditions, and features. Hence, the impact of the non-default parameters can be easily minimized and optimized as in other ML-based mechanisms.

VII. Discussion

We found that the traffic between the mining server and the client does not create heavy network data. Cryptojacking service providers are specially designed not to create as much communication as regular Internet communication. Table II and III show the difference between the packet per minute of cryptojacking malware and a regular Internet connection. In addition to this, after we evaluated our tests on the service providers we used [41], [50], we saw that average service providers use different IP addresses during the mining operation to communicate with the victims' client device and deliver the mining task via encrypted application data. This behavior of in-browser cryptojacking malware makes them harder to track inside busy networks. Moreover, it is also an indicator of the impracticability of IP and hosting-based static blacklisting implementations.

Another challenge we faced during our data collection process is the asymmetric communication between the client and malicious IP addresses. In a normal data communication process over the Internet, the data packets create some symmetry between the client and the server, but in the cryptojacking samples, there is no symmetry or pattern between the client and server. 88% of communication packets were produced by the client during the mining process and only 12% of packets were created by malicious servers. This asymmetric communication makes it harder to analyze the collected data and keep track of the communication pipelines between the client and the malicious server using different IP addresses. Therefore, in this paper, we investigated the detection of malicious device performing cryptomining instead of identifying the malicious IP. Identifying the malicious IP remains an open problem to be further investigated.

VIII. Related Work

After the surge of blockchain technologies and the widespread usage of cryptojacking malware, cryptojacking malware and detection methods drew the attention of academia as a new research area. The research in this area can be categorized under two main directions. The work proposed by Tekiner et al. [4] is a systematization of knowledge study on cryptojacking malware, which summarizes the techniques used by the cryptojacking malware and reviews the cryptojacking studies in the literature.

The first line of research in this area focused on analyzing different aspects of the cryptojacking malware ecosystem. In one of the first studies in the cryptojacking malware literature [59], the authors investigated Bitcoin miner samples in the binary format. The increase in the cryptojacking malware attacks and the price surge of cryptocurrencies in 2017 also attracted researchers' attention. Later, in [60], the authors analyzed 30k websites using coinhive libraries to mine Monero. The studies [61]–[63] analyzed the impact of cryptojacking malware on

the consumer devices and users. In other studies, the authors in [64] found a new attack type infected Microtik routers while the studies in [65], [66] performed campaign analysis of the cryptojacking samples.

The second line of research in this area focused on cryptojacking malware detection methods [16], [19]–[28], [67]. Researchers used both static and dynamic features to detect cryptojacking malware. The commonly used features are CPU events [17], [18], [20], [22], [23], [26], [68]–[72], memory activities [20], [22], [23], [67], JS compilation times [17], [68], and network traffic [20]–[24], [73]. These features are not feasible to be used to detect the cryptojacking malware targeting IoT devices because most of the IoT devices do not allow to be programmed to collect browser-specific features (e.g., JS compilation/execution time) or hardware-based feature (e.g., HPC). On the other hand, the network traffic-based features are ideal as the network traffic can be intercepted on the way. Previously, network traffic data has been used for malware detection in many studies. In [74], the authors used HTTPS traffic for the malicious client detection in an organization's network through the neural networks. Similarly, the authors in [75] used network traffic-based features to identify the unknown malware families. Other examples of the usage of network traffic-based features for malware detection are as follows: malicious app detection in Android [76]–[78], malware type detection [79]. On the other, some of the studies that used network traffic for the IoT malware detection are as follows: [80]–[82]. However, none of these studies specified the malware family. Compared to these studies in the literature, in our study, we used network traffic-based features to detect IoT cryptojacking malware.

Similar to ours, the studies that are using network traffic for cryptojacking detection are as follows: [20]–[24], [73]. The studies in [20], [22], [23] collect the aggregated network traffic features (e.g., the total number of bytes sent or received) from the browser together with other resource consumption features (e.g., memory). However, in our paper, we use only network traffic features as the other features can not be collected from IoT devices. On the other hand, the study in [24] collect the network data in an emulated environment while we deployed the miners on the real devices and collected both real malicious and benign network data. Finally, the closest studies to ours are [21], [73]. The study in [21] uses five, while the study in [73] uses six network traffic features based on the incoming and outgoing network traffic rates. Our paper differs from these studies in many ways. First of all, they do not focus on IoT devices; therefore, they do not have device diversity while we implemented a cryptominer even on a smart TV. Second, we tested with both in-browser (e.g., coinhive) and host-based (e.g., MinerGate) cryptojacking samples while both studies only experiment with host-based cryptojacking samples only. With this, we also tested varying throttle values that the attackers may use to see the potential attacker behavior strategies. Last, we used a time-series data automatic feature extraction tool, which created 788 different statistical features from its wide selection of statistical features and provided a near-perfect accuracy in general.

Cryptojacking detection studies can also be categorized as in-browser cryptojacking detection [5], [16], [19]–[28], host-based cryptojacking detection [19], [73], and cryptojacking type-agnostic detection studies [18], [67], [71], [72], [83]. As IoT devices can be targeted by both in-browser and host-based cryptojacking malware, in this paper, we aim to design a type-agnostic cryptojacking detection method.

IX. Conclusion

This paper proposed an accurate and efficient cryptojacking detection mechanism based on the features extracted from network traffic. Our mechanism was able to detect both in-browser and host-based cryptojacking malware. We achieved 99% detection accuracy with one-hour network traffic data used to train the machine learning classifier. We also designed novel attack scenarios to test our mechanism in attack configurations and home network settings. In addition, we also analyzed cryptojacking attacks on several different platforms to see the efficiency of our detection mechanism. We showed how different configurations that the attacker may use and different network settings that the mining would be performed on affect the detection accuracy. Moreover, we shared the network traffic we collected and the code publicly to accelerate the research in this area.

Acknowledgement

We would like to kindly thank the authors of [54] who graciously shared their malware samples. We also thank the anonymous reviewers for their feedback and time. This work was supported by the U.S. National Science Foundation (Award: NSF-CAREER CNS-1453647). The views expressed are those of the authors only, not of the funding agencies.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." 2008.
[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.
[3] "Official page of monero cryptocurrency and its white paper," https://www.getmonero.org/resources/research-lab/, accessed: 2020-02-16.
[4] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "Sok: Cryptojacking malware," in *6th IEEE European Symposium on Security and Privacy*, Virtual, September 2021.
[5] ——, "In-browser cryptomining for good: An untold story," in *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 2021, pp. 20–29.
[6] IBM-Security, "X-force threat intelligence index 2020," https://securityintelligence.com/series/ibm-x-force-threat-intelligence-index-2020, accessed: 2021-02-16.
[7] "Aws cloud-based cryptojacking report," https://www.cadosecurity.com/post/team-tnt-the-first-crypto-mining-worm-to-steal-aws-credentials, accessed: 2020-02-16.
[8] T. Seals, "Aws cloud-based cryptojacking report," https://threatpost.com/aws-cryptojacking-worm-cloud/158427/, accessed: 2021-2-23.
[9] Check-Point-Research, "Checkpoint 2020 cyber security report," https://research.checkpoint.com/2020/the-2020-cyber-security-report/, accessed: 2021-2-23.
[10] L. Greenemeier, "Crytojacking can corrupt the iot," https://www.scientificamerican.com/article/how-cryptojacking-can-corrupt-the-internet-of-things/, accessed: 2021-2-23.

[11] C. Cimpanu, "Mikrotik router hack affect 200k routers in the world," https://www.bleepingcomputer.com/news/security/massive-coinhive-cryptojacking-campaign-touches-over-200-000-mikrotik-routers/, accessed: 2021-2-23.

[12] "Trend micro iot botnet article," https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/into-the-battlefield-a-security-guide-to-iot-botnets, accessed: 2021-07-04.

[13] D. McMillen, "What is the mirai botnet?" https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/, accessed: 2020-11-02.

[14] S. R. Department, "Estimated iot device count by 2025," https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/, accessed: 2021-2-23.

[15] "Security intelligence, liquor iot botnet," https://securityintelligence.com/news/weekly-security-news-roundup-mirai-inspired-liquorbot-botnet-mining-for-monero/, accessed: 2021-07-04.

[16] R. Tahir, S. Durrani, F. Ahmed, H. Saeed, F. Zaffar, and S. Ilyas, "The browsers strike back: countering cryptojacking and parasitic miners on the web," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 703–711.

[17] I. Petrov, L. Invernizzi, and E. Bursztein, "Coinpolice: Detecting hidden cryptojacking attacks with neural networks," *arXiv:2006.10861*, 2020.

[18] G. Mani, V. Pasumarti, B. Bhargava, F. T. Vora, J. MacDonald, J. King, and J. Kobes, "Decrypto pro: Deep learning based cryptomining malware detection using performance counters," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2020, pp. 109–118.

[19] H. Darabian, S. Homayounoot, A. Dehghantanha, S. Hashemi, H. Karimipour, R. M. Parizi, and K.-K. R. Choo, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, pp. 1–11, 2020.

[20] J. D. P. Rodriguez and J. Posegga, "Rapid: Resource and api-based detection against in-browser miners," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 313–326.

[21] J. Z. i Muñoz, J. Suárez-Varela, and P. Barlet-Ros, "Detecting cryptocurrency miners with netflow/ipfix network measurements," in *2019 IEEE International Symposium on Measurements & Networking*, pp. 1–6.

[22] R. Ning, C. Wang, C. Xin, J. Li, L. Zhu, and H. Wu, "Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1873–1881.

[23] C. Kelton, A. Balasubramanian, R. Raghavendra, and M. Srivatsa, "Browser-based deep behavioral detection of web cryptomining with coinspy."

[24] H. N. C. Neto, M. A. Lopez, N. C. Fernandes, and D. M. Mattos, "Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking," *Annals of Telecommunications*, pp. 1–11, 2020.

[25] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, M. Yang, Y. Zhang, Z. Qian, and H. Duan, "How you get shot in the back: A systematical study about cryptojacking in the real world," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1701–1713.

[26] M. Musch, C. Wressnegger, M. Johns, and K. Rieck, "Web-based cryptojacking in the wild," *arXiv preprint arXiv:1808.09474*, 2018.

[27] W. Wang, B. Ferrell, X. Xu, K. W. Hamlen, and S. Hao, "Seismic: Secure in-lined script monitors for interrupting cryptojacks," in *European Symposium on Research in Computer Security*, 2018, pp. 122–142.

[28] 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*, pp. 1714–1730.

[29] "Webassembly," https://webassembly.org//, accessed: 2021-02-23.

[30] F. Naseem, A. Aris, L. Babun, E. Tekiner, and S. Uluagac, "MINOS: A lightweight real-time cryptojacking detection system," in *28th Annual Network and Distributed System Security Symposium, NDSS, February 21-25, 2021*, 2021.

[31] N. Avital, "New research: Crypto-mining drives almost 90% of all remote code execution attacks," https://www.imperva.com/blog/new-research-crypto-mining-drives-almost-90-remote-code-execution-attacks/, accessed: 2021-2-23.

[32] D. Olenick, "Miner into third party zoom," https://www.trendmicro.com/en_us/research/20/d/zoomed-in-a-look-into-a-coinminer-bundled-with-zoom-installer.html, accessed: 2020-04-13.

[33] M. Santos, "utorrent update smuggles shady cryptocurrency miner into your computer," https://99bitcoins.com/utorrent-update-cryptocurrency-miner/, accessed: 2020-03-31.

[34] C. McDonald, "Cryptojacking malware hid into emails," https://www.mailguard.com.au/blog/brandjacking-malware-hiding, accessed: 2021-02-23.

[35] B. G. Mark Vicente, Johnlery Triunfante, "Cve-2019-2725 exploited, used to deliver monero miner," https://www.trendmicro.com/en_ca/research/19/f/cve-2019-2725-exploited-and-certificate-files-used-for-obfuscation-to-deliver-monero-miner.html, accessed: 2021-2-23.

[36] D. McMillen and M. Alvarez, "Mirai iot botnet: Mining for bitcoins?" https://securityintelligence.com/mirai-iot-botnet-mining-for-bitcoins/, accessed: 2021-2-23.

[37] "Tsfresh official page," https://tsfresh.readthedocs.io/en/latest/.

[38] S. Labovitz, "Criteria for selecting a significance level: A note on the sacredness of. 05," *The American Sociologist*, pp. 220–222, 1968.

[39] "Tefresh relevance table official documents," https://tsfresh.readthedocs.io/en/latest/_modules/tsfresh/feature_selection/relevance.html, accessed: 2021-02-23.

[40] "Lg webos development framework," https://webostv.developer.lge.com/, accessed: 2020-02-16.

[41] "The official webpage of webmine," http://webmine.cz/, accessed: 2021-4-7.

[42] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," *arXiv preprint arXiv:2102.06249*, 2021.

[43] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side(-channel) of mobile devices: A survey on network traffic analysis," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2658–2713, 2018.

[44] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[45] "General explanation of port mirroring," https://www.miarec.com/faq/what-is-port-mirroring, accessed: 2021-06-04.

[46] C. Nachreiner, "Anatomy of an arp poisoning attack," *Retrieved July*, vol. 4, p. 2005, 2003.

[47] "Ettercap project official page," https://www.ettercap-project.org/, accessed: 2021-06-04.

[48] "Wireshark," https://www.wireshark.org/, accessed: 2021-07-21.

[49] "Wordpress official page," https://wordpress.com/, accessed: 2021-06-04.

[50] "The official webpage of webmine pool," https://www.webminepool.com/, accessed: 2021-4-7.

[51] "Official page of minergate web mining pool," https://www.minergate.com, accessed: 2020-04-13.

[52] "Implementation of randomx proof of work algorithm." https://github.com/tevador/RandomX.

[53] "Official page of antminingpool web mining pool," https://v3.antpool.com/home, accessed: 2020-04-13.

[54] "Labayen, víctor; magana, eduardo; morato oses, daniel; izal, mikel (2020), "network traffic for machine learning classification dataset"," http://doi.org/10.17632/5pmnkshffm.1, 2020.

[55] "Tsfresh official features page," https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html.

[56] M. J. Schervish, "P values: what they are and what they are not," *The American Statistician*, vol. 50, no. 3, pp. 203–206, 1996.

[57] "Official page of scikit-learn python library," https://scikit-learn.org/stable/, accessed: 2021-07-22.

[58] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[59] D. Y. Huang, H. Dharmdasani, S. Meiklejohn, V. Dave, C. Grier, D. McCoy, S. Savage, N. Weaver, A. C. Snoeren, and K. Levchenko, "Botcoin: Monetizing stolen cycles." in *Network and Distributed Systems Security (NDSS) Symposium*. Citeseer, 2014.

[60] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based cryptojacking," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2018, pp. 58–66.

[61] P. H. Meland, B. H. Johansen, and G. Sindre, "An experimental analysis of cryptojacking attacks," in *Nordic Conference on Secure IT Systems*. Springer, 2019, pp. 155–170.

[62] P. Papadopoulos, P. Ilia, and E. Markatos, "Truth in web mining: Measuring the profitability and the imposed overheads of cryptojacking," in *International Conference on Information Security (ISC)*. Springer, 2019, pp. 277–296.

[63] R. Holz, D. Perino, M. Varvello, J. Amann, A. Continella, N. Evans, I. Leontiadis, C. Natoli, and Q. Scheitle, "A retrospective analysis of user exposure to (illicit) cryptocurrency mining on the web," *arXiv preprint arXiv:2004.13239*, 2020.

[64] H. L. Bijmans, T. M. Booij, and C. Doerr, "Just the tip of the iceberg: Internet-scale exploitation of routers for cryptojacking," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 449–464.

[65] ——, "Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1627–1644.

[66] S. Pastrana and G. Suarez-Tangil, "A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth," in *Proceedings of the Internet Measurement Conference (IMC)*, 2019, pp. 73–86.

[67] M. Conti, A. Gangwal, G. Lain, and S. G. Piazzetta, "Detecting covert cryptomining using hpc," *arXiv preprint arXiv:1909.00268*, 2019.

[68] A. Kharraz, Z. Ma, P. Murley, C. Lever, J. Mason, A. Miller, N. Borisov, M. Antonakakis, and M. Bailey, "Outguard: Detecting in-browser covert cryptocurrency mining in the wild," in *The World Wide Web Conference (WWW)*, 2019, pp. 840–852.

[69] A. D. Yulianto, P. Sukarno, A. A. Warrdana, and M. Al Makky, "Mitigation of cryptojacking attacks using taint analysis," in *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*. IEEE, 2019, pp. 234–238.

[70] W. Bian, W. Meng, and M. Zhang, "Minethrottle: Defending against wasm in-browser cryptojacking," in *Proceedings of The Web Conference (WWW) 2020*, 2020, pp. 3112–3118.

[71] N. Lachtar, A. A. Elkhail, A. Bacha, and H. Malik, "A cross-stack approach towards defending against cryptojacking," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 126–129, 2020.

[72] D. Tanana, "Behavior-based detection of cryptojacking malware," in *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*. IEEE, 2020, pp. 0543–0545.

[73] M. Caprolu, S. Raponi, G. Oligeri, and R. Di Pietro, "Crypto mining makes noise," *arXiv:1910.09272*, 2019.

[74] P. Prasse, L. Machlica, T. Pevnỳ, J. Havelka, and T. Scheffer, "Malware detection by analysing encrypted network traffic with neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 73–88.

[75] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 134–142.

[76] M. Zaman, T. Siddiqui, M. R. Amin, and M. S. Hossain, "Malware detection in android by network traffic analysis," in *2015 international conference on networking systems and security (NSysS)*. IEEE, 2015, pp. 1–5.

[77] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Information Sciences*, vol. 433-434, pp. 346–364, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025517307077

[78] A. Arora and S. K. Peddoju, "Minimizing network traffic features for android mobile malware detection," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, 2017, pp. 1–10.

[79] M. Piskozub, F. De Gaspari, F. Barr-Smith, L. Mancini, and I. Martinovic, "Malphase: Fine-grained malware detection using network flow data," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 774–786.

[80] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "Iot malware network traffic classification using visual representation and deep learning," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 444–449.

[81] V. Rey, P. M. S. Sánchez, A. H. Celdrán, G. Bovet, and M. Jaggi, "Federated learning for malware detection in iot devices," *arXiv preprint arXiv:2104.09994*, 2021.

[82] N. Guizani and A. Ghafoor, "A network function virtualization system for detecting malware in large iot based networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1218–1228, 2020.

[83] A. Ahmad, W. Shafiuddin, M. N. Kama, and M. M. Saudi, "A new cryptojacking malware classifier model based on dendritic cell algorithm," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–5.

X. Appendix

In this section, we give the detailed results for the experiments in Section VI-D.

	Attack Case	Benign	Malicous
Scenario 1	Server	1217322	1240922
	Desktop	234272	238996
	IoT	64064	70664
Scenario 2	Aggressive	1520681	1537036
	Robust	26669	29010
	Stealthy	9880	13459
Scenario 3	In-Browser	305874	314906
	Host-Based	1251356	1229399
Scenario 4	Fully compromised (Overall)	1557230	1561443
Scenario 5	Partially compromised (IoT + Laptop)	246853	247143
Scenario 6	Single compromised (IoT)	13459	11745
Scenario 7	IoT compromised (IoT + IoT)	47978	57205

TABLE XVI: Dataset sizes for all scenarios (Scenario 1-7) with our own benign dataset described in Table XI.

	Attack Case	Accuracy	Precision	Recall	F1 Score	Test_roc
Scenario 1	Server	0.99	0.99	0.99	0.99	0.99
	Desktop	0.95	0.95	0.95	0.95	0.98
	IoT	0.95	0.95	0.95	0.95	0.98
Scenario 2	Aggressive	0.93	0.93	0.93	0.93	0.93
	Robust	0.95	0.95	0.95	0.95	0.95
	Stealthy	0.88	0.88	0.88	0.87	0.92
Scenario 3	In-Browser	0.95	0.95	0.95	0.95	0.98
	Host-Based	0.97	0.97	0.97	0.97	0.99
Scenario 4	Fully compromised (Overall)	0.98	0.98	0.98	0.98	0.99
Scenario 5	Partially compromised (IoT + Laptop)	0.97	0.97	0.97	0.97	0.99
Scenario 6	Single compromised (IoT)	0.97	0.97	0.97	0.97	0.99
Scenario 7	IoT compromised (IoT + IoT)	0.95	0.95	0.95	0.95	0.98

TABLE XVII: Classification results of all scenarios (Scenario 1-7) with our own benign dataset described in Table XI.

	Classifier (SVM)			Accuracy	Precision	Recall	F1-Score	Test_ROC
	Kernel	C	Gamma					
Scenario 10	Linear	1	Scale	1.0	1.0	1.0	1.0	1.0
	Poly	1	Scale	0.83	0.83	0.83	0.83	0.92
	RBF	1	Scale	0.83	0.84	0.83	0.83	0.91
	Sigmoid	1	Scale	0.72	0.72	0.72	0.72	0.76
	Linear	1	Auto	1.0	1.0	1.0	1.0	1.0
	Poly	1	Auto	0.88	0.88	0.88	0.88	0.93
	RBF	1	Auto	0.66	0.80	0.66	0.61	0.70
	Sigmoid	1	Auto	0.52	0.27	0.52	0.35	0.5
	Linear	2	Scale	1.0	1.0	1.0	1.0	1.0
	Poly	2	Scale	0.84	0.84	0.84	0.84	0.82
	RBF	2	Scale	0.87	0.88	0.87	0.87	0.92
	Sigmoid	2	Scale	0.73	0.73	0.73	0.73	0.76
	Linear	2	Auto	1.0	1.0	1.0	1.0	1.0
	Poly	2	Auto	0.88	0.88	0.88	0.88	0.93
	RBF	2	Auto	0.66	0.80	0.66	0.61	0.70
	Sigmoid	2	Auto	0.52	0.27	0.52	0.35	0.50

TABLE XVIII: Results of the non-default parameters experiments described in Section VI-D3.