CJ-Sniffer: Measurement and Content-Agnostic Detection of Cryptojacking Traffic

Yebo Feng, Jun Li, Devkishen Sisodia
{yebof,lijun,dsisodia}@cs.uoregon.edu
University of Oregon
Eugene, Oregon, USA

ABSTRACT

With the continuous appreciation of cryptocurrency, *cryptojacking*, the act by which computing resources are stolen to mine cryptocurrencies, is becoming more rampant. In this paper, we conduct a measurement study on cryptojacking network traffic and propose CryptoJacking-Sniffer (CJ-Sniffer), an easily deployable, privacy-aware approach to protecting all devices within a network against cryptojacking. Compared with existing approaches that suffer from privacy concerns or high overhead, CJ-Sniffer only needs to access anonymized, content-agnostic metadata of network traffic from the gateway of the network to efficiently detect cryptojacking traffic. In particular, while cryptojacking traffic is also cryptocurrency mining traffic, CJ-Sniffer is the first approach to distinguishing cryptojacking traffic from *user-initiated* cryptocurrency mining traffic, making it possible to only filter cryptojacking traffic, rather than blindly filtering all cryptocurrency mining traffic as commonly practiced. After constructing a statistical model to identify all the cryptocurrency mining traffic, CJ-Sniffer extracts variation vectors from packet intervals and utilizes a long short-term memory (LSTM) network to further identify cryptojacking traffic. We evaluated CJ-Sniffer with a packet-level cryptomining dataset. Our evaluation results demonstrate that CJ-Sniffer achieves an accuracy of over 99% with reasonable delays.

CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → *Network measurement*; **Network security**; **Network monitoring**; • **Information systems** → Data mining.

KEYWORDS

cryptocurrency, cryptomining, cryptojacking, anomaly detection, network traffic analysis

ACM Reference Format:
Yebo Feng, Jun Li, Devkishen Sisodia. 2022. CJ-Sniffer: Measurement and Content-Agnostic Detection of Cryptojacking Traffic. In *25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2022), October 26–28, 2022, Limassol, Cyprus.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3545948.3545973

1 INTRODUCTION

With the frenzy of the cryptocurrency market, cryptocurrency mining (cryptomining) has become a method of making huge profits. In fact, cryptomining is critical in many blockchain-based systems, as it not only provides a means to verify cryptocurrency transactions, but more importantly, also helps establish consensus through different mechanisms [32], such as Proof of Work (PoW) and Proof of Space (PoS). Therefore, to encourage cryptomining, cryptocurrency systems usually reward miners with transaction fees and extra coins. Unfortunately, the lucrative potential of cryptomining has caught the attention of hackers, who compromise personal computers, servers, or even Internet-of-Things (IoT) devices, such as smart TVs, to mine cryptocurrencies (e.g., BTC, XMR) [51]. Such activity is called **cryptojacking**, which is the unauthorized use of someone else's computing resources to mine cryptocurrency. Such a hacker is also called a **cryptojacker** and such a resource is said to be **cryptojacked**.

There are many methods of conducting cryptojacking [24]. For example, a cryptojacker can trick a victim into clicking on a malicious link in an email to download cryptomining scripts onto their computer, infect a website with JavaScript code to automatically run the code by a victim's browser when it visits the website, or compromise a server to stealthily execute cryptomining programs in the background. Although Coinhive, an in-browser mining service provider, was shut down in March 2019, cryptojacking has still been active and evolving [44]. According to the Unit 42 Cloud Threat Report [43], from December 2020 to February 2021, 17% of organizations with a cloud infrastructure showed signs of cryptojacking, causing significant computing resource abuse and tremendous economic loss. Recently, the SophosLabs team also disclosed that cryptojackers compromised unpatched Exchange servers to mine XMR [10].

There have been many endpoint-based approaches to cryptojacking defense [14, 24, 46]. By monitoring software operations, website visits, or hardware conditions, an endpoint-based approach can often achieve decent accuracy and is usually easy to deploy if resources permit, but it is almost infeasible to deploy it on all endpoints due to user inertia and many endpoints are indeed resource-constrained. To address this issue, researchers and developers proposed network-based approaches to detect general cryptomining activities by analyzing cryptominers' network traffic [11, 26, 37]. These approaches are often deployed only at a network gateway, avoiding the need for every device in the network to deploy a defense solution. However, as application-layer behavior inference through layer-3 network traffic is extremely difficult, whether these network-based approaches are based on IP blocklists or deep packet inspection, or conduct state-of-the-art traffic analysis, they can hardly distinguish cryptojacking from **user-initiated cryptomining**, which is cryptomining performed by legitimate users of computing devices in use. Thus, network-based approaches treat all cryptocurrency mining traffic as either legitimate and allowing all of the traffic, or malicious and dropping all of the traffic.

To fill the missing gap, in this paper, we first thoroughly measure and model the characteristics of cryptomining network traffic, including exploring the tiny differences between cryptojacking traffic and user-initiated cryptomining traffic. We then propose CryptoJacking-Sniffer (CJ-Sniffer), a content-agnostic, easily deployable approach that not only can detect cryptomining activities, but also can distinguish cryptojacking from user-initiated cryptomining, only by analyzing layer-3 network traffic.

CJ-Sniffer operates in three phases. In the first phase, CJ-Sniffer inspects packet sizes of every connection to discard obviously irrelevant connections, which could significantly reduce the data volume for further analysis. In the second phase, CJ-Sniffer detects cryptomining connections by comparing every connection's packet interval distribution with labeled traffic. In the third phase, CJ-Sniffer leverages the long short-term memory (LSTM) model [23] to distinguish cryptojacking connections from user-initiated cryptomining connections. As a result, CJ-Sniffer can efficiently classify network connections into three categories: cryptojacking connections, user-initiated cryptomining connections, and other connections.

Our work makes the following contributions:

- CJ-Sniffer as a content-agnostic, network-based approach to detecting cryptojacking activities is not only efficient since it does not inspect packet payload, but also preserves user privacy by only leveraging a limited amount of anonymized metadata of packets.
- CJ-Sniffer is the first work to distinguish cryptojacking traffic from user-initiated cryptomining traffic. This is important because it is often necessary to only filter cryptojacking traffic while preserving user-initiated cryptomining traffic from a device. While the difference between cryptomining traffic and other network traffic is already challenging to detect at a network gateway, the difference between cryptojacking traffic and user-initiated cryptomining traffic is significantly harder to detect.
- CJ-Sniffer is easy and efficient to deploy. While cryptojacking is rampant, any institution, enterprise, or an individual user can deploy CJ-Sniffer at their network's gateway to safeguard all the computing devices in their network from cryptojacking at line speed.
- We collect, measure, and release the first labeled, packet-level cryptomining traffic dataset to the research community. It contains more than 500 hours of both cryptojacking and user-initiated cryptomining traffic from various types of computing devices.

We evaluated CJ-Sniffer with real-world network traffic and found that its efficacy and efficiency are both high. Even running on an ordinary personal computer, CJ-Sniffer is capable of providing real-time traffic monitoring for an enterprise or campus-level network. To reach a detection accuracy of 95%, it only needs to collect approximately 160 network packets from a cryptojacked device, and approximately 200 network packets to reach an accuracy of 97%, all with nearly zero false alarms.

The rest of this paper is organized as follows. After we outline related work in Section 2, we describe our discoveries of cryptojacking and cryptomining traffic properties (Section 3). We then illustrate CJ-Sniffer's design in Section 4 and evaluate it in Section 5. We discuss its limitations and open issues in Section 6 and conclude the paper in Section 7. Of a particular note here is that the research described in this paper is extended from our early poster paper published in [20].

2 RELATED WORK

In the past few years, researchers have proposed various cryptomining or cryptojacking defense approaches. According to the deployment position of these approaches, they are generally either endpoint-based or network-based, or a combination of the two with multiple deployment positions.

2.1 Endpoint-based Cryptojacking Defense

Endpoint-based cryptojacking defense is dominated by various endpoint-based cryptojacking detection approaches. Usually, these approaches can be easily embedded in antivirus tools, system firewalls, or even browsers.

Endpoint-based cryptojacking detection approaches can use different types of input to conduct detection. The most common input is source code of software or websites. For example, CMTracker [24] first uses hash-based and stack-based profiling methods to extract features from websites and then matches the features against handcrafted rules to identify websites with cryptojacking code; SEISMIC [46] derives semantic signatures from known cryptojacking scripts and then matches running scripts at an endpoint against signatures to detect cryptojacking scripts. Besides, hardware conditions can also be used as input. Gomes et al. [21] extracted features from the CPU usage and used machine learning to detect cryptojacking processes. Tahir et al. [42] proposed a machine learning detection solution based on features from Hardware Performance Counters (HPCs) values. However, hardware-based approaches may generate an excess of false alarms, as many legitimate processes sometimes utilize hardware in a way similar to cryptojacking. To address this issue, CoinPolice [38] actively changes the execution speed of processes, then collects execution traces at various execution speeds, and inputs the collected data into a deep neural network to pinpoint cryptojacking scripts.

These approaches can often achieve a decent accuracy because they have access to a variety of system and software metrics from end-users. However, the popularity of these approaches is limited by user habits, as not all users are willing to install cryptojacking detection software and have it monitor their computer thoroughly. Also, some IoT devices, such as smart furniture and health-monitoring devices, have too little resources to deploy these approaches. Additionally, cryptojacking malware is becoming more sophisticated; for example, hackers could obfuscate their code to thwart an endpoint-based approach. It is thus necessary to update an endpoint-based approach frequently.

Table 1: Comparisons of selected network-based cryptomining detection approaches (○: not support; ◐: partially support; ●: fully support).

Approach	Content Agnostic	Cryptomining Detection	User-initiated Cryptomining v.s. Cryptojacking
DPI-based solution	○	●	●
Cisco solution [8]	◐	●	○
Munoz et al. [26]	●	●	○
Pastor et al. [37]	●	●	○
Hu et al. [25]	●	●	○
MineHunter [50]	●	●	○
CJ-Sniffer	●	●	●

2.2 Network-based Cryptomining Defense

Network-based cryptomining defense operates at certain vantage points of a network so that any devices within the network can be protected. During the early rise of cryptomining, some companies, schools, and institutions deployed some simple network-based approaches to guard their computing devices. For example, network administrators can block the IP addresses of confirmed mining pools and websites hosting cryptomining code [8, 49]; or, intrusion detection systems can conduct deep packet inspection (DPI) to discover specific cryptomining text strings in packet payloads [16]. These approaches are easy to develop and deploy, but they can only offer preliminary defense against cryptomining due to their low levels of accuracy.

Later, as network traffic analysis techniques [36] evolved, several projects began to detect cryptomining activities with content-agnostic traffic flows. Munoz et al. [26] inspected network flows in NetFlow or IPFIX format with some learning-based algorithms (e.g., SVM, CART, C4.5, and Naïve Bayes) to detect cryptomining activities. Caprolu et al. [12] built a random-forest-based framework to classify network traffic related to pool mining, solo mining, and activities from active full nodes. Pastor et al. [37] extracted several features from NetFlow data and leveraged deep learning models to detect encrypted cryptomining malware connections; Hu et al. [25] indicate that using random forest with extracted discriminative network traffic features can accurately and efficiently detect cryptomining traffic; MineHunter [50] detects cryptomining connections by analyzing packet intervals with a similarity calculation algorithm based on credible probability estimation. These network-based approaches come with many advantages: (1) once deployed in a network, they can protect all users inside the network; (2) they do not analyze message content from users, thus protecting user privacy; (3) they have a higher throughput than DPI methods due to their smaller input size (network flow records vs. packet trace with payloads).

However, none of the aforementioned approaches distinguish between *cryptojacking* and *user-initiated cryptomining* activities (as demonstrated in Table 1). They cannot meet the desire of many networks to forbid only cryptojacking traffic but still allow the user-initiated cryptomining traffic.

2.3 Hybrid Cryptojacking Defense

Hybrid cryptojacking defense combines both endpoint-based and network-based approaches. For example, Gomes et al. introduced CryingJackpot [22], which first extracts features from both network traffic flows and endpoint operating system logs and then utilizes unsupervised machine learning algorithms to discover cryptojacked devices with a high F1-Score. Although hybrid cryptojacking defense can achieve robust and accurate detection results on different types of cryptojacking activities, it requires a variety of data to operate, making it challenging to deploy in the real world.

3 MEASUREMENT OF CRYPTOMINING AND CRYPTOJACKING TRAFFIC

As discussed in Section 1, we classify cryptomining activities into two categories: (1) *user-initiated cryptomining*, refers to the cryptomining conducted by the owner or legitimate user of the computing device; (2) *Cryptojacking*, refers to the cryptomining conducted by attackers using stolen computing resources. In this section, we study and measure the network traffic generated from both types of cryptomining activities. The design and evaluation of CJ-Sniffer is based on these measurement results.

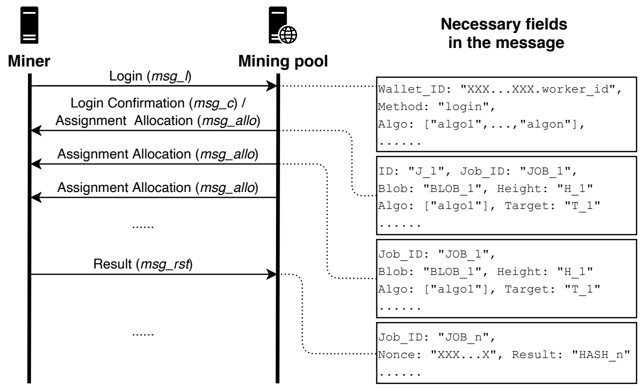
We mainly investigate the mining traffic related to XMR [30], which dominates the cryptojacking [9] campaigns in today's Internet. XMR applies CryptoNight [31] as its hash function, which is ASIC resistance and enables cryptojackers to obtain reasonable profits through different types of computing devices [27]. Conversely, other cryptocurrencies are much harder to mine with ordinary computing devices, making them less profitable for cryptojackers. In addition, other cryptocurrencies that employ PoW consensus mechanisms, will generate similar cryptomining traffic to that of XMR. We further demonstrate this point in Section 5.3 and show that by studying the cryptomining traffic of XMR, we can detect many other PoW-based cryptomining traffic

3.1 In-depth analysis of general cryptomining traffic

In PoW-based cryptocurrency systems, cryptominers need to participate in puzzle solving competitions to obtain rewards or transaction fees. Due to the exceptional difficulty of solving these puzzles, individual miners with limited computing capacities, particularly the cryptojacked devices, have to join mining pools to ensure stable and prompt profits [33]. Thus, the traffic between miners and the mining pool is the key to detecting cryptomining.

For coordination, the mining pool and miners need to obey certain protocols, such as Stratum [40], to register nodes, distribute tasks, and submit results. Figure 1 illustrates the mining process and messages between miners and the mining pool. No matter what types of protocols the mining pool utilizes, there should be at least four types of messages to cover the necessary mining operations:

- The login or registration message msg_l, enabling miners to join the mining pool, can have 75 to 600 bytes per message.
- The login confirmation message msg_c, confirming the login status, sometimes comes with an assignment allocation message.
- The assignment allocation message msg_{allo}, allocating the most recent mining task to the miner, should at least have 285 bytes.
- The result message msg_r, returning calculated results to the mining pool, usually has more than 200 bytes.


Necessary fields in the message
Miner
Mining pool
Login (*msg_l*)
Login Confirmation (*msg_c*) / Assignment Allocation (*msg_allo*)
Assignment Allocation (*msg_allo*)
Assignment Allocation (*msg_allo*)
......
Result (*msg_rst*)
......

```
Wallet_ID: "XXX...XXX.worker_id",
Method: "login",
Algo: ["algo1",...,"algon"],
......
```

```
ID: "J_1", Job_ID: "JOB_1",
Blob: "BLOB_1", Height: "H_1"
Algo: ["algo1"], Target: "T_1"
......
```

```
Job_ID: "JOB_1",
Blob: "BLOB_1", Height: "H_1"
Algo: ["algo1"], Target: "T_1"
......
```

```
Job_ID: "JOB_n",
Nonce: "XXX...X", Result: "HASH_n"
......
```

Figure 1: Network messages generated by cryptomining and their necessary fields.

Besides, the login message and confirmation message often appear only once during each connection. Thus, assignment allocation messages dominate the inbound traffic and result messages dominate the outbound traffic during the whole cryptomining process. Figure 2 illustrates communication between a miner and mining pool. Excluding TCP control packets such as SYN and ACK packets, we can see that the inbound traffic is mainly composed of assignment allocation messages and the outbound traffic is mainly composed of result messages.

The size of the packet in the cryptomining traffic is distinctive. Figure 3 illustrates packet size ranges of cryptomining traffic and some contrast traffic (collected from our lab and campus network). We can see that the sizes of cryptomining packets are more monotonous compared with others, since the same type of cryptomining messages usually have similar lengths. In addition, cryptomining packets generally have smaller sizes compared with other types of traffic. For most web applications, their maximum packet sizes are usually subject to the network's maximum transmission unit (MTU), which is usually around 1500 bytes. However, the information in each cryptomining operation cannot fill even half of the MTU. Besides, due to timeliness requirements, the miner or mining pool cannot bank messages and send them in a single packet. Thus, all the cryptomining packets are relatively small in size.

Moreover, the frequency of inbound packets is more stable than that of outbound packets. This is because mining tasks expire quickly with the growth of the blockchain. The mining pool needs to keep sending assignment allocation messages (inbound packets) with the growth of the blockchain to ensure that miners always have valid up-to-date tasks. In the short term, the speed of blockchain expansion is very stable, therefore the frequency at which assignment allocation messages (inbound packets) are generated is also stable. On the other hand, the frequency of result submission messages (outbound packets) is related to the hash rate—the speed at which a device is completing an operation in the cryptomining code. The higher the computing performance of a device, the higher its hash rate, and the higher the frequency it sends result messages (outbound packets) to the mining pool. Figure 4 illustrates this pattern, where the frequency of outbound packets is proportional to the hash rate while the frequency of inbound packets is relatively stable. It's also important to note that the frequency of assignment allocation messages doesn't need to be higher than that of result submission messages. Because one assignment can usually derive more than one sub-results to fully complete.

Last but not least, the generation of cryptomining packets is not subject to human behavior. The time to generate the next result message depends on the time the device completes the hash backtracking. Analogously, the time to generate the next assignment allocation message depends on the time the mining pool proposes a new task. Hence, the intervals of cryptomining packets exhibit stable and unique distributions. These interval distributions can be treated as fingerprints of cryptomining traffic (illustrated in Figure 5 and 6). We can identify a cryptomining connection by checking whether these interval distributions are obeyed.

3.2 Cryptojacking *vs.* user-initiated cryptomining traffic

After studying cryptojacking activities, we found that they differ from user-initiated cryptomining in the *robustness* of hash rate. This further results in a difference in the network traffic generated by the two.

```
<script src="https://www.XXXpool.com/lib/base.js"></script>
<script>
    var miner=WMP.User('<your-site-key>','<username>',{
        threads: 4,  // number of maximum threads
        autoThreads: true, // adjust the number of threads automatically
        throttle: 0.85, // maximum system load
        forceASMJS: false
    });
    miner.start();
</script>
```

Listing 1: JavaScript code piece of web-based cryptojacking.

Unlike user-initiated cryptomining, cryptojacking activities have the following features that can lead to unstable mining hash rate: (1) First of all, as injected programs, the execution priority of cryptojacking scripts is usually low, making them easy to be disturbed by the resource manager of operating systems. Modern operating systems will not blindly allocate a huge chunk of resources to random processes (e.g., cryptojacking) because they need to ensure there is enough redundant load to handle high-priority tasks that may occur. Conversely, user-initiated cryptomining usually runs with a high execution priority by system administrators. Even with many background processes, user-initiated cryptomining still has access to a large amount of hardware and software resources. (2) Cryptojackers use stolen computing resources to mine cryptocurrencies. To prevent being discovered by legitimate users of the computing device, cyptojackers usually conduct cryptojacking in surreptitious ways. For example, cryptojackers may dynamically adjust the hash rate of cryptojacked devices to prevent interference with the normal use of users. Listing 1 shows part of the JavaScript code that is used for browser-based cryptojacking. We can see that the hacker lets the device automatically adjust the number of mining threads to make sure the system load is under 85%. By doing

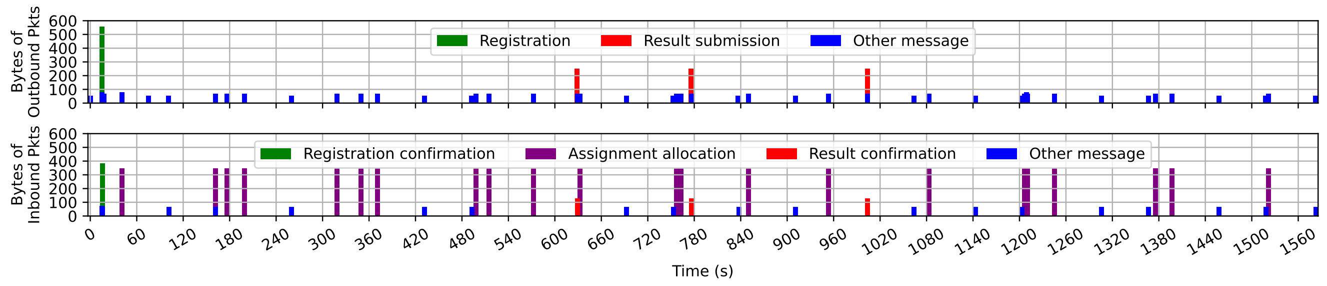


Figure 2: Visualized network packets between an XMR mining pool and an XMR miner. Other messages refer to packets that do not carry meaningful payloads, such as TCP SYN and FIN packets.

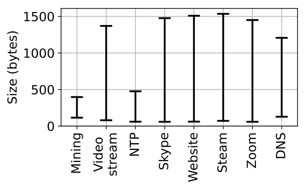


Figure 3: Range of packet sizes of different types of traffic.

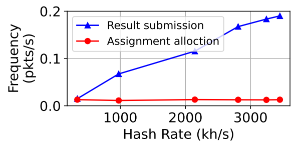
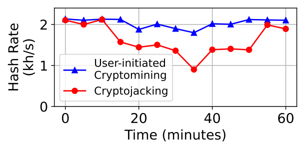
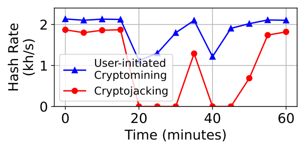


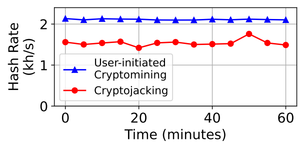
Figure 4: Frequency of different types of cryptomining packets under different hash rates (captured using i7 8700k CPU and XMRig [2]).



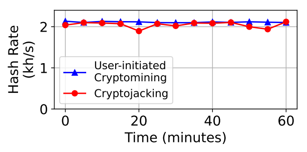
(a) For most cryptojacking activities, the hash rate exhibits irregular changes due to automatic adjustments of computing resources.



(b) Due to minimum memory requirements or time-based resource throttling, hash rates of some cryptojacking activities go to zero from time to time.



(c) Some cryptojacking activities have stable hash rates. However, these hash rates are still lower than that of user-initiated cryptomining using the same hardware.



(d) In rare cases, the attacker utilizes all computing resources of the cryptojacked device, generating hash rates that are similar to user-initiated cryptomining.

Figure 7: Trends in hash rate of different types of cryptojacking activities (extracted from our collected dataset described in Table 2).

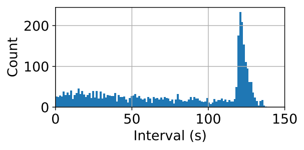


Figure 5: Interval distribution of assignment allocation messages (inbound packets larger than 285 bytes).

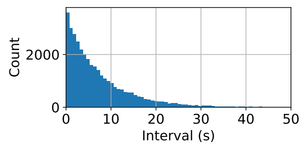


Figure 6: Interval distribution of result submission messages (outbound packets larger than 200 bytes).

so, legitimate users will not sense any abnormality in the computing device, thereby allowing the cryptojacking program to run in the background for a long time. Otherwise, the user will quickly sense the abnormality and scan the device for a virus. (3) Moreover, executions of cryptojacking scripts usually rely on executions of existing software running in the system such as the browser, terminal, or Apache server. Due to the uncertainty of human behavior, the execution situation of such software is inconstant, making the computing resources devoted to cryptomining erratic.

We measured the hash rate trends of user-initiated cryptomining and cryptojacking in the real world on the same machines (demonstrated in Figure 7). The measurement results are consistent with our previous analysis that cryptojacking produces unstable hash rates in most cases. Furthermore, according to measurements in Section 3.1, this hash-rate instability will generate result submission messages in unstable frequencies.

4 DESIGN OF CJ-SNIFFER

In this section, we describe the design details of CJ-Sniffer, an in-network-based cryptojacking traffic detection approach.

CJ-Sniffer can be deployed within the intrusion detection system (IDS) of any router or switch between the mining pools and devices to be protected. Particularly, CJ-Sniffer functions effectively at the gateway of a network, since this vantage point enables CJ-Sniffer to access complete inbound and outbound traffic from all the devices in the network. In addition, people can treat CJ-Sniffer as a cloud service and stream the network traffic to it for detection. To prevent leakage of user information in this case, the IP addresses in the traffic flows are anonymized with Crypto-PAn [48] and sent encrypted through a Kafka message queue. Thus, the CJ-Sniffer

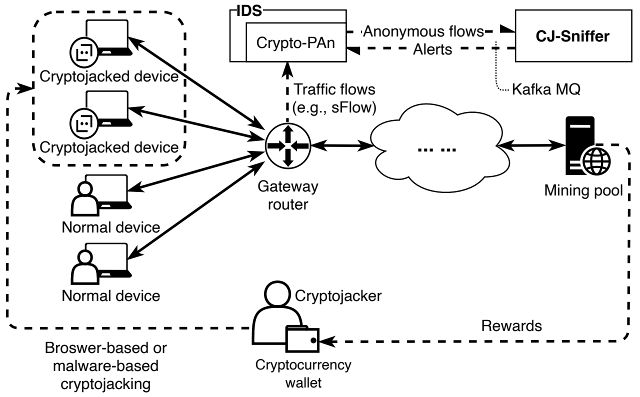


Figure 8: One operational model of CJ-Sniffer, where it is deployed apart from the IDS.

service provider or any third parties cannot fetch necessary information to trace back to individuals in the network. Figure 8 illustrates the operational model of CJ-Sniffer in this type of deployment,

As stated in Section 1, CJ-Sniffer detects cryptojacking activities in three phases. The first detection phase can quickly filter out irrelevant traffic flows, leaving only suspicious ones for future analysis. The second phase inputs suspicious traffic and outputs confirmed cryptomining traffic. At last, the third detection phase utilizes an LSTM model to distinguish cryptojacking traffic from user-initiated cryptomining traffic.

4.1 Preprocessing

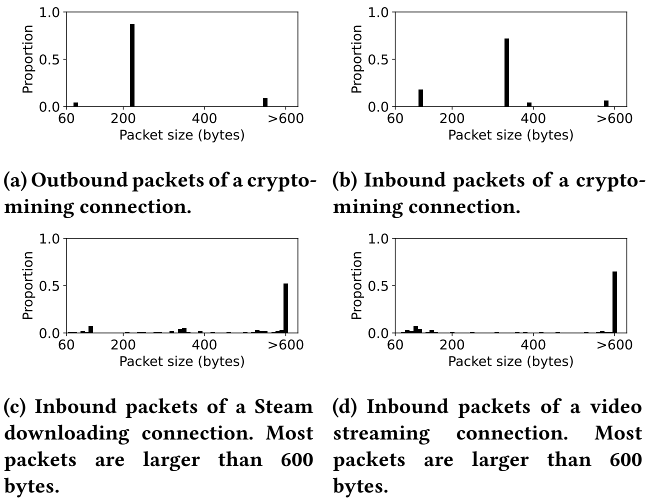
CJ-Sniffer requires the timestamps and six fields in the IP packet headers for detection, which are the source and destination IP addresses, the source and destination port numbers, the protocol type, and the packet size. CJ-Sniffer is therefore content-agnostic, as it does not require any payload information.

To obtain this content-agnostic data, we recommend installing sFlow [39] in the router or switch to stream traffic flows to CJ-Sniffer in real time. Other network traffic capturing engines like Netmap [41] and PF-RING [7] are also compatible with the proposed approaches. Once CJ-Sniffer fetches the traffic flows, it extracts the aforementioned data fields and stores them in a table for future analysis. In the table, each entry represents a received packet. Meanwhile, CJ-Sniffer discards all other information from the traffic flows.

4.2 Phase one: rapid filtration

In phase one, CJ-Sniffer rapidly filters out irrelevant network traffic and picks out only suspicious traffic for future analysis. This step can significantly reduce the size of the traffic data for inference, increasing the throughput of CJ-Sniffer.

To conduct rapid filtration, CJ-Sniffer first eliminates packets without payload (e.g., TCP SYN, ACK, and FIN packets), packets of irrelevant protocols (e.g., ICMP), and internal packets. CJ-Sniffer then groups remaining packets by connections, which are defined as consecutive packets that are sent between the same IP addresses and port numbers. Then, CJ-Sniffer inspects the packet sizes in each



(a) Outbound packets of a cryptomining connection.

(b) Inbound packets of a cryptomining connection.

(c) Inbound packets of a Steam downloading connection. Most packets are larger than 600 bytes.

(d) Inbound packets of a video streaming connection. Most packets are larger than 600 bytes.

Figure 9: Packet size distribution of different types of traffic.

connection to judge whether it could be a suspicious cryptomining connection.

We define a sliding time window to monitor each connection and make a judgment. During each time window t, CJ-Sniffer collects a list of packets P ($P = \{p_1, p_2, p_3, ..., p_n\}$) from the ongoing connection. It then represents inbound and outbound packets with their size values and stores them into two lists (l_{in} for inbound packets and l_{out} outbound packets) respectively. Once this time window is about to end, CJ-sniffer will check the value distributions of l_{in} and l_{out} to determine whether the connection is suspicious. According to the traffic measurement study in Section 3.1, CJ-Sniffer utilizes three sets of rules to determine suspicious connections:

- The majority of the packets' sizes should lie within the cryptomining packet size range;
- The majority of the outbound packets' sizes should be uniform (illustrated in Figure 9a);
- The majority of the inbound packets' sizes should have values drawn from the same narrow interval (illustrated in Figure 9b).

Once packets within a connection follow these three rules, CJ-Sniffer will label this connection as suspicious and pass it onto the next phase for deeper analysis. Note, labeling a connection as suspicious does not mean this connection is confirmed to be related with cryptomining. For example, some connections generated by NTP or DNS can have similar distribution of packet sizes. Hence, CJ-Sniffer only filters out obviously irrelevant traffic in this phase to accelerate the detection process.

4.3 Phase two: detection of cryptomining

In phase two, CJ-Sniffer uses filtered network traffic from phase one as input and outputs detected cryptomining traffic. According to the cryptomining traffic study in Section 3.1, to accurately identify cryptomining traffic, CJ-Sniffer inspects the packet intervals of suspicious connections to determine whether they are generated

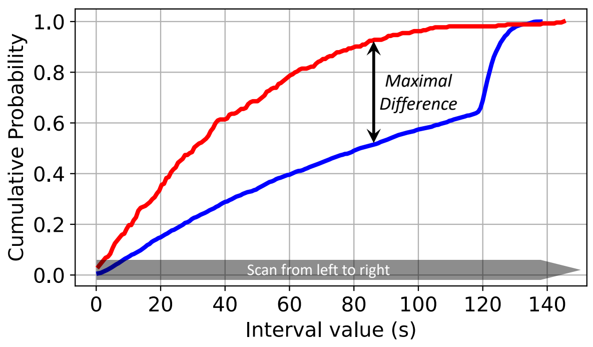


Figure 10: Illustration of the two-sample Kolmogorov–Smirnov statistic. The blue line corresponds to the empirical distribution function of labeled cryptomining traffic. The red line corresponds to the empirical distribution function of downloading traffic. The black arrow is the two-sample KS statistic.

by the cryptominer or the mining pool. Specifically, CJ-Sniffer compares both the inbound and outbound packet interval distributions to distributions from collected cryptomining traffic. As long as one of the inbound or outbound packet intervals follow the same distribution with cryptomining traffic data, CJ-Sniffer will label the connection as a cryptomining connection.

The distribution compliance test is conducted with the Two-Sample Kolmogorov–Smirnov (KS) test [34], which is a nonparametric testing approach to determine whether two data samples come from the same distribution. Compared with other approaches to testing the distribution compliance, the KS test has no restrictions on the size of data sample, which means little cryptomining traffic data can help achieve decent accuracy. Moreover, the KS test is distribution-free. Users can easily update the contrast sample to cover the latest cryptomining traffic regardless of the sample's distribution.

The Two-Sample KS test works as follows. Suppose that the inbound or outbound packet interval sample from P has size m with an observed cumulative distribution function of $F(x)$. Furthermore, suppose that the labeled cryptomining sample Q has size n with an observed cumulative distribution function of $G(x)$. CJ-Sniffer defines the null hypothesis (H_0) as: both samples come from a population with the same distribution. It also defines the Two-Sample KS statistic $D_{m,n}$ with Equation 1 (illustrated in Figure 10).

$$D_{m,n} = \max_x |F(x) - G(x)|. \tag{1}$$

After calculating $D_{m,n}$, CJ-Sniffer rejects the null hypothesis at significance level α if $D_{m,n} > D_{m,n,\alpha}$, where $D_{m,n,\alpha}$ is the critical value and can be calculated with Equation 2.

$$\begin{aligned} D_{m,n,\alpha} &= c(\alpha)\sqrt{\frac{n+m}{n \cdot m}} \\ &= \sqrt{-\ln(\frac{\alpha}{2}) \cdot \frac{1+\frac{m}{n}}{2m}}. \end{aligned} \tag{2}$$

Conversely, if $D_{m,n} \le D_{m,n,\alpha}$, CJ-Sniffer will accept the null hypothesis H_0 and label the incoming traffic as cryptomining.

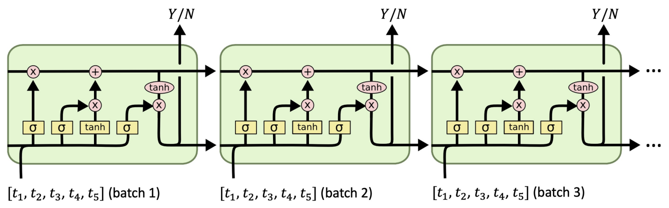
The significance level α in the Two-Sample KS test is the probability of rejecting the null hypothesis when it is true. Users of CJ-Sniffer can adjust the value of α to reach different detection sensitivities. A large significance level α can lead to a small critical value $D_{m,n,\alpha}$, which will raise the standard of the distribution compliance test. In our implementation, we set α as 0.10, which is a relatively large value but can increase the usability of CJ-Sniffer by reducing the false positive rate.

Algorithm 1 Cryptomining traffic detection using KS test.

```
1:  Input: $P$, $m$, $Q$, $G(x)$, $n$, $k$, $\alpha$    ▷ $P$ is the packet list with $m$ packets, $Q$ is the labeled cryptomining packet list with $n$ packets, $G(x)$ is the cumulative distribution function of $Q$, $k$ is the granularity for calculating the KS statistic, $\alpha$ is the significance level
2:  Output: 1 for cryptomining traffic, 0 for other traffic
3:  $l_P = inboundInterval(P)$    ▷ extract the inbound packet intervals and store them in a list
4:  $l_Q = inboundInterval(Q)$
5:  $r = max(l_Q) - min(l_Q)$    ▷ calculate the range of $G(x)$
6:  initialize list $l_d$    ▷ to store the differences of two cumulative distribution functions (CDFs)
7:  for $i$ in $range(k)$ do
8:      $x \longleftarrow \frac{i \cdot r}{k} + min(l_Q)$
9:      $l \longleftarrow \{j | j \in l_P \text{ and } j \le x\}$
10:     $f \longleftarrow \frac{l.size()}{l_P.size()}$    ▷ calculate the CDF value at $x$ for $P$
11:     append $|f - G(x)|$ to $l_d$
12:     if $f == 1$ then
13:         break
14:     end if
15: end for
16: $D_{m,n} \longleftarrow max(l_d)$
17: if $D_{m,n} \le \sqrt{-\ln(\frac{\alpha}{2}) \cdot \frac{1+\frac{m}{n}}{2m}}$ then
18:     return 1    ▷ accept the hypothesis $H_0$
19: else
20:     return 0
21: end if
```

Algorithm 1 demonstrates the detailed procedure that CJ-Sniffer utilizes to determine cryptomining traffic. CJ-Sniffer only tests the distribution compliance of inbound packet intervals, as inbound packet intervals are more robust compared with outbound packet intervals (discussed in Section 3). Moreover, to reduce the process time, the cumulative distribution function of labeled cryptomining traffic is calculated beforehand. Therefore, when receiving new suspicious traffic, CJ-Sniffer only needs to build one cumulative distribution function.

Once CJ-Sniffer completes the analysis in phase two, network operators can choose the next step according to their needs. For instance, if some companies and institutions prohibit any cryptomining inside their networks, then they can stop at this phase and block any connection that is labeled as cryptomining. Meanwhile, some network operators allow user-initiated cryptomining activities. Nonetheless, they still want to distinguish cryptojacking traffic



Y/N
Y/N
Y/N
σ
σ
tanh
σ
tanh
$[t_1, t_2, t_3, t_4, t_5]$ (batch 1)
$[t_1, t_2, t_3, t_4, t_5]$ (batch 2)
$[t_1, t_2, t_3, t_4, t_5]$ (batch 3)

Figure 11: Structure of the LSTM model that CJ-Sniffer utilizes.

to safeguard users' computing resources. In this case, CJ-Sniffer can enter phase three to dig further into the labeled cryptomining traffic.

4.4 Phase three: detection of cryptojacking

In the third phase, CJ-Sniffer uses detected cryptomining traffic as input and outputs identified cryptojacking traffic. According to the detection results of CJ-Sniffer, network operators can conduct access control only on cryptojacking connections while still leaving user-initiated cryptomining connections alive.

Enlightened from the measurement results in Section 3.2, CJ-Sniffer distinguishes cryptojacking traffic from user-initiated cryptomining traffic by inspecting the long-term robustness of the result submission messages. For various reasons, the hash rate of cryptojacking activities is relatively unstable compared with user-initiated cryptomining. This hash rate instability further affects the frequency of result submission messages (msg_r). The key to identifying cryptomining connections is to recognize such frequency instability.

To achieve the goal, CJ-Sniffer utilizes a LSTM machine learning model to learn the cryptojacking traffic patterns, with cryptojacking traffic as positive samples and user-initiated cryptomining traffic as negative samples. Then, CJ-Sniffer applies the trained LSTM model to identify cryptojacking traffic. LSTM is a type of artificial recurrent neural network (RNN) architecture used in the field of deep learning [23]. Compared with other candidate approaches, LSTM is particularly suitable to detect cryptomining traffic for the following reasons: (1) as an RNN variant, LSTM is good at processing time series or sequential data, such as cryptomining traffic; (2) LSTM introduces both a short-term and a long-term memory component, allowing it to uncover hidden frequency changes out of traffic data from a long-term perspective; (3) LSTM is a learning-based approach, which means it can automatically and dynamically learn the detection thresholds from our collect dataset without any human interventions. Nonetheless, the detection component here is modular. Users may use other machine learning algorithms or statistical approaches to fit their environments once the inputs are the same.

As for the input data, CJ-Sniffer extracts the variation vector v from outbound traffic to profile the changes in result submission message frequency. CJ-Sniffer first picks out only the outbound packets from the traffic and divides them into batches. Every batch consists of six consecutive packets with five consecutive packet intervals. Then, CJ-Sniffer generates a variation vector v to represent each batch of data, where $v = [t_1, t_2, t_3, t_4, t_5]$ and t_n denotes the interval of two consecutive packets. CJ-Sniffer will input the variation vectors into the LSTM model and conduct cryptojacking detection batch by batch.

Table 2: Information of the collected cryptomining dataset.

Cryptocurrencies	Size	Length	Mining Chips
XMR, ETH	250 MB	~ 750 hours	Intel Core i5-5257U, Intel Core i7-6820HQ, Apple M1, AMD Ryzen 5 1400 quad-core, NVIDIA GeForce GTX 1080, Intel Core i7-8700K, Intel Xeon CPU E5-2430, AMD Radeon RX 570, AMD Ryzen 5 1400 quad-core + AMD Radeon RX 570.

Figure 11 illustrates the structure of the LSTM model that CJ-Sniffer utilizes. As the input data is not complicated, CJ-Sniffer employees a Vanilla LSTM model, which has a single hidden layer of LSTM units, and an output layer used to make the decision. We set the number of neurons in the hidden layer to 20 according to a rule of thumb that was introduced in [17]. Equation 3 demonstrates the rule, where N_i denotes the number of input neurons, N_o denotes the number of output neurons, N_s denotes the number of samples in the training set, α is an arbitrary scaling factor (usually ranges from 2 to 10), and N_h denotes the maximum number of neurons in the hidden layer.

$$N_h = \frac{N_s}{\alpha \cdot (N_i + N_o)}. \tag{3}$$

Moreover, the LSTM uses binary cross entropy as the loss function and Sigmoid as the activation function for the output neuron, as this combination is the most commonly used for binary classification problems. CJ-Sniffer keeps inputting variation vectors as time-series data to the LSTM, until the incoming traffic is identified as cryptojacking or all the batches have been processed.

The LSTM model requires training before it can be used. Since there are no existing cryptojacking traffic datasets available in public repositories, we captured both user-initiated cryptomining traffic and cryptojacking traffic to train the LSTM model. We also release part of the packet-level dataset with this paper.

5 EVALUATION

We evaluated CJ-Sniffer in campus network environments with real-world network traffic. In this section, we first describe how we collect labeled cryptomining traffic and real-world contrast traffic (Section 5.1). Then, we show our evaluation results of CJ-Sniffer regarding the efficacy of cryptomining traffic detection (Section 5.2), the ability of detecting other cryptocurrencies' mining traffic (Section 5.3), the efficacy of cryptojacking traffic detection (Section 5.4), and comparisons with other network-based approaches (Section 5.5). In the end, we evaluate the operation efficiency and the real-world deployability of CJ-Sniffer (Section 5.6.2).

5.1 Data collection

To support the measurement study of cryptomining traffic, train the statistical model and the machine learning model, and promote related research, we collected a labeled cryptomining traffic dataset from multiple calculation platforms, including both servers and personal computers, CPUs and GPUs. The cryptomining traffic dataset contains both user-initiated cryptomining traffic and cryptojacking traffic.

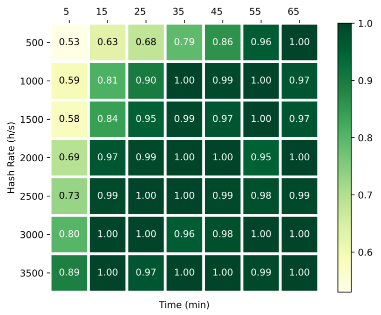


Figure 12: Accuracy scores of detecting cryptomining traffic with different hash rates and length.

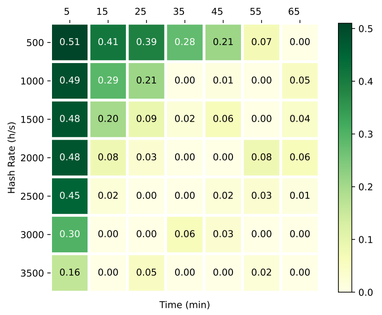


Figure 13: False negative rates of detecting cryptomining traffic with different hash rates and length.

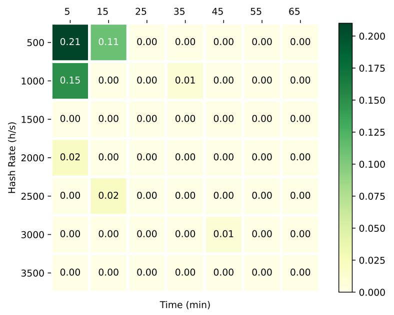


Figure 14: False positive rates of detecting cryptomining traffic with different hash rates and length.

Table 3: Information of the collected contrast network traffic.

Network	Size	Length	Types of Traffic Contained
Campus (10 Gbps), Lab (1 Gbps)	316 GB	~ 35,100 connections, ~ 2,000 hours in total	**Traffic without obvious periodic regularities:** Media streaming, VoIP service, HTTP, SMTP, FTP, Skype, Zoom, Gaming, Tunneling and proxy services, etc. **Traffic with obvious periodic regularities:** NTP, DNS, Control services of IoT appliances, Notification services, STUN, Google static content, etc.

To collect the dataset, we installed both user-initiated cryptomining and cryptojacking software on different computing devices in different network environments (e.g., home, lab, campus, etc.). We also created several websites with cryptojacking scripts from WebMinePool [5], CoinIMP [1], Easy Pool Miner [3], and Minero [4]. Then, we used Wireshark [35] to capture the packet-level traffic along with labeling information generated by these cryptomining activities. Table 2 shows the basic information of the collected dataset. The development and evaluation of CJ-Sniffer is based on this dataset. Moreover, we release a subset of the dataset to public [18], which has around 550 hours of cryptomining traffic, with all the noise and human-behavior-related traffic removed due to ethical considerations.

Besides, we captured around 316 GB of contrast traffic from our lab network (link bandwidth: 1 Gbps) and campus network (link bandwidth: 10 Gbps) [1]. Table 3 shows the basic information of the collected contrast traffic. By using port-based and graphlet-based [29] traffic classification approaches, we classified the contrast network traffic into several categories (e.g., HTTP, NTP, SMTP, etc.). Furthermore, according to the traffic patterns, we divided these categories into two groups. One is traffic without obvious periodic regularities, which is easier to distinguish from cryptomining traffic. The other is traffic with obvious periodic regularities, which is more difficult to distinguish from cryptomining traffic. The collected contrast traffic basically cover most types of network traffic we can see from medium-sized companies/institutions. We later mixed all the contrast traffic with the cryptomining traffic to evaluate the detection accuracy of CJ-Sniffer.

[1] We anonymously collected the contrast traffic and omitted all the private content data before storage. We have also obtained the Institutional Review Board (IRB) approval for the traffic collection.

5.2 Efficacy of cryptomining traffic detection

To evaluate CJ-Sniffer's efficacy of detecting general cryptomining traffic, we divided the dataset into two parts—the training set and testing set. The training set is used to train the detection model, containing around 200 hours of labeled cryptomining traffic. The testing set is treated as the input of CJ-Sniffer to evaluate its efficacy. As cryptomining traffic and other types of traffic are unbalanced in reality, the ratio of contrast traffic to cryptomining traffic is more than 20:1 in our testing set. Besides, to evaluate the detection accuracy in different scenarios, we further divided the testing set into seven groups according to the cryptomining hash rates.

Figure 12 demonstrates the accuracy scores of detecting cryptomining connections using CJ-Sniffer. Figure 13 demonstrates the false negative rates of detecting cryptomining traffic using CJ-Sniffer; We can see that detecting cryptomining traffic of different hash rates or lengths will derive totally different accuracy and false negative scores. As a statistics-based cryptomining detection approach, CJ-Sniffer can achieve a better efficacy with a larger data sample. Under this scenario, cryptomining traffic with longer durations or larger hash rates is easier to be detected by CJ-Sniffer, since these sets of traffic contain more interval samples for analysis. In general, to reach an accuracy of more than 0.95 in detecting both low-hash-rate and high-hash-rate cryptomining traffic, CJ-Sniffer needs to collect around 160 network packets generated from the device in each processing unit. If the cryptojacked devices are all high-performance devices (with hash rates of more than 2000h/s), they only need 15 minutes to generate this many network packets, which is significantly quicker compared with MineHunter that needs around 2 hours of traffic to achieve a similar efficacy.

Figure 13 demonstrates the false negative rates of detecting cryptomining traffic using CJ-Sniffer. Similar to accuracy scores, CJ-Sniffer needs shorter time to detect high-hash-rate devices' cryptomining traffic. Once CJ-Sniffer collects 55 minutes of traffic in each processing unit, it can achieve zero false negative rates for detecting both low-hash-rate and high-hash-rate cryptomining traffic.

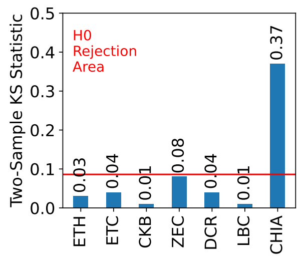


Figure 15: Two-sample KS statistics between XMR and other cryptocurrencies' mining traffic.

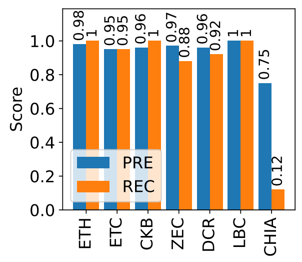


Figure 16: CJ-Sniffer's precision and recall scores in identifying other cryptocurrencies' mining traffic.

5.3 Adaptability to other cryptocurrencies

CJ-Sniffer is mainly built upon XMR mining data. However, cryptomining/cryptojacking traffic may be associated with other cryptocurrencies. In this section, we examine the adaptability of CJ-Sniffer. We collected several hours of cryptomining traffic of other cryptocurrencies (i.e., ETH, ETC, BTC, DCR, LBC, ZEC, and CHIA). The data volume may be insufficient for thorough measurement but is enough for testing. We then evaluated the ability of CJ-Sniffer (trained with XMR traffic) in detecting such cryptomining traffic.

The test is conducted on normalized traffic data. Figure 15 demonstrates the two-sample KS statistics between XMR and other cryptocurrencies' cryptomining packet intervals. The closer these values are to 0, the more similar these traffic intervals are to XMR's, and the more likely CJ-Sniffer can detect such traffic. From the results, we can see that cryptomining packet intervals of ETH, ETC, BTC, DCR, LBC, and ZEC come from the same distribution as XMR's, because their KS test statistics are too small to reject the H_0 hypothesis. Conversely, CHIA, possibly due to the adoption of a different consensus mechanism (i.e., Proof of Space), is quite different from XMR regarding cryptomining traffic. The detection evaluation is consistent with the statistics (Figure 16). CJ-Sniffer achieves decent precision and recall scores in detecting almost all the PoW-based cryptomining traffic.

5.4 Efficacy of cryptojacking traffic detection

CJ-Sniffer observes the hash rate robustness of ongoing cryptomining traffic to distinguish cryptojacking from user-initiated cryptomining. Therefore, the efficacy of cryptojacking traffic detection may be influenced by the background process running on the device. Here, the background process refers to all the computing processes executed by the legitimate user of the cryptojacked device. Theoretically, if the background processes cause a high system load, the computing resource allocated to cryptojacking programs will be volatile, leading to obviously unstable hash rates and becoming easier to be detected by CJ-Sniffer. However, when the background processes only cause a little system load, the system is more likely to allocate stable computing resources to cryptojacking programs, even if they are in low priorities. Therefore, CJ-Sniffer may fail to detect such cryptojacking activities.

Table 4: Efficacy of cryptojacking traffic detection with different system loads of background processes.

	System load of background processes.				
	0%	10%	20%	30%	40%
Accuracy	0.4714	0.5467	0.6176	0.7244	0.9390
Precision	0.8750	0.9130	0.9231	0.9348	0.9770
Recall	0.0875	0.2413	0.3673	0.5181	0.9140
FPR	0.0167	0.0317	0.0417	0.0411	0.0282
FNR	0.9125	0.7586	0.6327	0.4819	0.0860
F1 Score	0.1591	0.3818	0.5255	0.6667	0.9444
	System load of background processes.				
	50%	70%	80%	90%	100%
Accuracy	0.9630	0.9877	0.9877	0.9938	0.9890
Precision	0.9770	0.9888	0.9891	0.9890	0.9897
Recall	0.9551	0.9888	0.9891	1	0.9897
FPR	0.0274	0.0135	0.0143	0.0143	0.0118
FNR	0.0449	0.0112	0.0109	0.0000	0.0103
F1 Score	0.9659	0.9888	0.9891	0.9945	0.9897

We evaluate the efficacies of cryptojacking traffic detection with different background processes and present the results in Table 4. From the table, we can see that CJ-Sniffer can achieve good accuracy scores and false positive rates in any situation. However, we also notice that it can only reach decent recall scores and false negative rates when the loads of background processes are higher than 30%. Otherwise, CJ-Sniffer may have difficulty distinguishing cryptojacking traffic from user-initiated cryptomining traffic. Fortunately, in these scenarios, the computing device is in idle time, thereby limiting the effect such cryptojacking activities have on legitimate processes. Therefore, a cryptojacking attack is less of a concern under such circumstances. Still, CJ-Sniffer can tell network administrators that these traffic are cryptomining traffic in all the cases.

5.5 Comparison evaluation

In this subsection, we compare CJ-Sniffer with four other representative solution: 1. Cisco's commercial solution [8] that integrated in routers; 2. SVM-based approach proposed by Munoz et al. [26]; 3. Naïve Bayes-based approach proposed by Munoz et al. [26]; 4. MineHunter [50].

5.5.1 Comparison of cryptomining traffic detection. We use similar evaluation approaches in Section 5.2 to conduct the comparison evaluations for cryptomining traffic detection. The only difference is that each testing unit contains around 2 hours of traffic, so approaches that require a large amount of traffic (i.e., MineHunter and Munoz et al.) can achieve the best efficacy. Figure 17 illustrates receiver operating characteristic (ROC) curves generated by the five

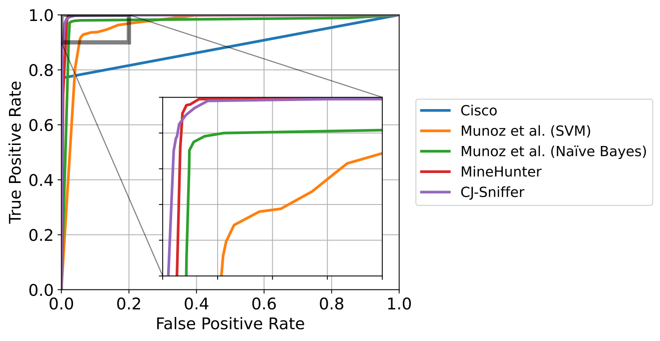
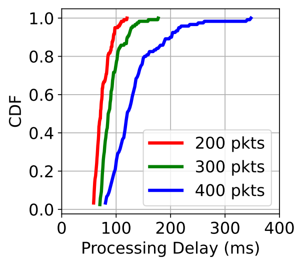
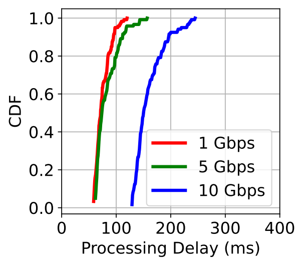


Figure 17: ROC curves of selected approaches for cryptomining traffic detection.



(a) Cumulative distribution functions (CDFs) of processing delays with different numbers of packets in each processing unit (link bandwidth: 1 Gbps).



(b) Cumulative distribution functions (CDFs) of processing delays when monitoring links with different bandwidths (processing unit: 200 pkts).

Figure 18: Processing delays of CJ-Sniffer under different operation circumstances (different link bandwidths & sizes of processing units).

approaches. We can see that CJ-Sniffer and MineHunter perform obviously better than approaches proposed by Munoz et al. and Cisco. CJ-Sniffer and MineHunter achieve similar accuracies, while CJ-Sniffer is better in false-positive rates, and MineHunter is better in true-positive rates.

As for the processing delay, all these five approaches can output the detection results within 2 seconds. Therefore, all of them can meet the velocity requirements in cryptomining traffic detection.

5.5.2 Comparison of cryptojacking traffic detection. Among the five approaches, CJ-Sniffer is the only one that can distinguish cryptojacking traffic from user-initiated cryptomining traffic. Other approaches simply treat these two groups of traffic as the same. Hence, CJ-Sniffer allows network operators to conduct more elastic and flexible security managements. For example, a network administrator can only filter cryptojacking traffic while preserving user-initiated cryptomining traffic from a device, which is a common requirement for many Virtual Private Servers (VPS) providers. For networks with strict usage restrictions (e.g., campus networks, enterprise networks, etc.), network administrators can simply utilize the first two phases of CJ-Sniffer to identify all cryptomining traffic.

5.6 System Efficiency

In this section, we evaluate the efficiency of CJ-Sniffer by time complexity analysis and measuring the system's processing delay in real-world environments.

5.6.1 Time complexity. Assume the total number of input packets is n. The time complexity of the *preprocessing* module is $O(n)$, as CJ-Sniffer simply receives network traffic from traffic capture engines and extracts necessary attributes from each packet header.

Then, CJ-Sniffer moves to *phase one*. CJ-Sniffer goes through each packet's extracted information to filter out noise and group them by connections, which takes $O(n)$ time to complete. After that, assume there are an ($0 \leq a \leq 1$) packets left, CJ-Sniffer still needs $O(an)$ time to complete rapid filtration. As a result, CJ-Sniffer's time complexity for phase one is $O(n + an)$.

In *phase two*, assume there are bn ($0 \leq b \leq 1$) packets left. Since the labeled samples' cumulative distribution function $G(x)$ is prebuilt, CJ-Sniffer can utilize Algorithm 1 to detect cryptomining traffic, allowing it to complete the analysis in $O(kbn)$ time.

In *phase three*, CJ-Sniffer employees an LSTM model to detect cryptojacking traffic. The LSTM model performs detections batch-by-batch and needs $O(1)$ time to process each batch. Assumes there are cn ($0 \leq c \leq 1$) batches, CJ-Sniffer needs $O(cn)$ to complete phase three.

Overall, CJ-Sniffer takes $O((a + kb + c)n)$ time to detect cryptojacking traffic from captured network traffic, as it conduct all the detection processes sequentially. In addition, k is a constant and a, b, c are numbers between zero and one. Therefore, CJ-Sniffer's time complexity is $O(n)$, where n is the total number of input packets.

5.6.2 Processing delay. We then measured the processing delay of CJ-Sniffer in the different real-world environments. Here, the processing delay refers to the time it takes for CJ-Sniffer to output the detection result after the data collection is completed. According to the design of CJ-Sniffer, there are two factors that can affect the processing delay: (1) the number of packets in each processing unit, and (2) the bandwidth of the link that CJ-Sniffer is monitoring. We changed both of the factors to evaluate the efficiency of CJ-Sniffer. We used a personal computer with an i7 8700k 4.7-GHz CPU and 32-GB memory to conduct the evaluation. The network traffic was ported from the gateway routers of our campus network and lab network.

Figure 18a demonstrates the delay with different number of packets in each processing unit. CJ-Sniffer takes 89 milliseconds on average to output the result when there are 200 packets per unit. If the number increases to 300, the processing delay just slightly increases to around 100 milliseconds. Although there is a noticeable performance drop when the number of packets in each processing unit reaches 400, according to evaluations in Section 5.2, 200 packets per detection unit is enough to reach a deployable accuracy.

Figure 18b demonstrates the delay with different link bandwidths. We can see that with a personal computer's computing power, CJ-Sniffer is able to monitor a campus-level or medium-company-level network (10 Gbps) and finish processing the data in less than 0.25 seconds.

Therefore, the processing time data from the real deployment of CJ-Sniffer supports the aforementioned time complexity analysis result. CJ-Sniffer's monitoring and detection can reach line speed for most companies, campuses, or institutes, with affordable computing power.

6 DISCUSSION

A primary contribution of CJ-Sniffer is to use content-agnostic data to detect cryptomining traffic and further distinguish cryptojacking traffic from user-initiated cryptomining traffic in the network, which can protect users' privacy and increase the efficiency during the detection process. However, CJ-Sniffer still has a few open issues. In this section, we discuss these open issues, indicating possible limitations of CJ-Sniffer and raising several avenues for future research.

6.1 Bypassing CJ-Sniffer

CJ-Sniffer is essentially a detection approach based on traffic analysis. In the last two decades, researchers and developers have proposed a variety of countermeasures to bypass traffic analysis (e.g., packet padding [28], dummy packets [45], traffic morphing [47], etc.). However, cryptojackers cannot directly modify their cryptomining traffic to escape our detection system, because current cryptomining activities rely highly on public mining pools that require miners to strictly follow public mining protocols (i.e., Stratum [40]) and remove unnecessary content in packets to prevent virus injection. Hence, any modification of cryptomining traffic would possibly cause disconnections to the mining pool. On the other hand, bypassing CJ-Sniffer by creating a private mining pool with a newly designed cryptomining protocol is also unlikely. Due to intense computing power competition [13] among mining pools, private mining pools' hash rates are significantly less than commonly-used public mining pools, making private mining pools inefficient in mining new coins. Still, it is feasible for cryptojackers to leverage third parties to help hide their cryptomining traffic (e.g., proxy, VPN, Tor with traffic obfuscation [19]), thereby bypassing CJ-Sniffer. Although we have not observed this defense design in current cryptojacking software, investigating cryptojacking traffic detection in the presence of traffic tunneling is a promising future research direction.

6.2 Adaptability to other network environments

CJ-Sniffer's adaptability in different network environments can significantly affect its usability because re-training the detection model is a time-consuming process for learning-based approaches. Fortunately, according to our collected data in different network environments and investigations of the cryptomining mechanism, changes in cryptomining traffic are independent of different network environments. Therefore, a cryptomining detection model trained in one network environment can be directly applied to another network environment if the input traffic is of the same format (e.g., sFlow, in our implementation). However, blockchain systems may evolve by upgrading their mining mechanisms, such as the Arrow Glacier upgrade of ETH [6] and the CryptoNight V7 upgrade of XMR [15]. These upgrades can bring changes to the cryptomining traffic patterns, thereby making detection models trained by previous data no longer effective. To tackle this issue, researchers need to continue enriching and updating the cryptomining traffic dataset so that the dataset can cover most recent cryptomining traffic patterns.

6.3 Direct hash rate inference

CJ-Sniffer detects cryptojacking by identifying hash rate fluctuations. A more straightforward method is to directly identify the specific hash rates from the traffic, which not only can help detect cryptojacking, but also can estimate the amount of computing power that is devoted to cryptomining. However, identifying specific hash rates is more challenging than identifying hash rate fluctuations, especially when there is no content data involved. In the future, extending CJ-Sniffer with more traffic features, preprocessing steps, or training data so that it can discover more information from content-agnostic traffic data is a promising research direction.

7 CONCLUSION

Cryptojacking is becoming far more sophisticated and threatening than before. To tackle this problem, we propose CJ-Sniffer, a privacy-aware cryptojacking detection approach that only relies on content-agnostic network traffic to conduct detections. CJ-Sniffer applies a three-phase procedure to identify cryptojacking traffic. It first filters out obviously irrelevant traffic to increase the detection throughput. It then accurately detects cryptomining traffic by conducting distribution compliance tests on packet intervals. Lastly, CJ-Sniffer digs deeper into the packet interval patterns, utilizing an LSTM model to distinguish cryptojacking traffic from user-initiated cryptomining traffic.

By introducing traffic analysis into cryptojacking detection, CJ-Sniffer is able to safeguard computing resources with superior efficiency and deployability. With a personal computer and traffic flow access to the network gateway, CJ-Sniffer is able to provide real-time traffic monitoring for a company-level network. More importantly, the privacy of users will not be violated throughout the detection process. In addition, unlike other network-based solutions that simply treat all types of cryptomining activities the same, CJ-Sniffer can distinguish cryptojacking traffic from user-initiated cryptomining traffic through their subtle differences. Therefore, CJ-Sniffer can provide hierarchical detection results, allowing network operators to conduct more elastic security management.

We have evaluated CJ-Sniffer with real-world network traffic and found that its efficacy and efficiency are both high. With the computing power of an i7 8700k 4.7-GHz CPU and 32-GB memory, CJ-Sniffer is able to achieve an accuracy of over 99% for PoW-based cryptocurrency systems with reasonable delays when monitoring a campus-level network.

ACKNOWLEDGMENTS

This material is based upon work partially supported by Ripple under the University Blockchain Research Initiative (UBRI). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Ripple.

We thank Zhiqian Zhao for her valuable contribution to the cryptojacking traffic generation and collection.

REFERENCES

[1] 2017. CoinIMP: FREE JavaScript Mining - Browser Mining. https://www.coinimp.com/. Accessed: 2021-04-15.

[2] 2017. XMRig: high performance, open source, cross platform XMR miner. https://github.com/xmrig/xmrig. Accessed: 2021-04-15.

[3] 2018. Easy Pool Miner. https://jefreesujit.github.io/easyminer/. Accessed: 2021-04-15.

[4] 2019. Minero: Monero miner for Web browsers. https://minero.cc/. Accessed: 2021-04-15.

[5] 2019. WebMinePool: Multifunctional mining service for site owners and individuals. https://www.webminepool.com/. Accessed: 2021-04-15.

[6] 2021. The history of Ethereum. https://ethereum.org/en/history/. Accessed: 2022-4-6.

[7] 2021. PF-RING: High-speed packet capture, filtering and analysis. https://www.ntop.org/products/packet-capture/pf_ring/. Accessed: 2021-05.

[8] Spenser Reinhardt Lilia Gonzalez Medina Josh Reynolds Alan Smith Alex Mcdonnell, Nichols Mavis. 2019. Blocking Cryptocurrency Mining Using Cisco Security Products.

[9] Hugo LJ Bijmans, Tim M Booij, and Christian Doerr. 2019. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1627–1644.

[10] Andrew Brandt. 2021. Compromised Exchange server hosting cryptojacker targeting other Exchange servers. https://news.sophos.com/en-us/2021/04/13/compromised-exchange-server-hosting-cryptojacker-targeting-other-exchange-servers/.

[11] Maurantonio Caprolu, Simone Raponi, Gabriele Oligeri, and Roberto Di Pietro. 2019. Cryptomining makes noise: a machine learning approach for cryptojacking detection. *arXiv preprint arXiv:1910.09272* (2019).

[12] Maurantonio Caprolu, Simone Raponi, Gabriele Oligeri, and Roberto Di Pietro. 2021. Cryptomining makes noise: Detecting cryptojacking via Machine Learning. *Computer Communications* 171 (2021), 126–139.

[13] Lin William Cong, Zhiguo He, and Jiasun Li. 2021. Decentralized mining in centralized pools. *The Review of Financial Studies* 34, 3 (2021), 1191–1235.

[14] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza M Parizi, and Kim-Kwang Raymond Choo. 2020. Detecting Cryptomining Malware: a Deep Learning Approach for Static and Dynamic Analysis. *Journal of Grid Computing* (2020), 1–11.

[15] dEBRYUNE, dnaleor, and Monero project. 2018. PoW change and key reuse. https://www.getmonero.org/2018/02/11/PoW-change-and-key-reuse.html. Accessed: 2022-4-6.

[16] Darragh Delaney. 2018. How to Detect Cryptocurrency Mining Activity on Your Network. https://www.netfort.com/blog/detect-cryptocurrency-mining-activity/.

[17] Howard B Demuth, Mark H Beale, Orlando De Jess, and Martin T Hagan. 2014. *Neural network design.* Martin Hagan.

[18] Yebo Feng. 2022. Packet-Level Cryptomining Network Traffic Dataset. https://github.com/yebof/CJ-Sniffer-Dataset.

[19] Yebo Feng. 2022. Toward Finer Granularity Analysis of Network Traffic. (2022).

[20] Yebo Feng, Devkishen Sisodia, and Jun Li. 2020. Poster: Content-agnostic identification of cryptojacking in network traffic. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 907–909.

[21] Fábio Gomes and Miguel Correia. 2020. Cryptojacking Detection with CPU Usage Metrics. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–10.

[22] Gilberto Gomes, Luis Dias, and Miguel Correia. 2020. CryingJackpot: Network flows and performance counters against cryptojacking. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–10.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[24] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Haixin Duan. 2018. 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*.

[25] Xiaoyan Hu, Zhuozhuo Shu, Xiaoyi Song, Guang Cheng, and Jian Gong. 2021. Detecting Cryptojacking Traffic Based on Network Behavior Features. In *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 01–06.

[26] Jordi Zayuelas i Muñoz, José Suárez-Varela, and Pere Barlet-Ros. 2019. Detecting cryptocurrency miners with NetFlow/IPFIX network measurements. In *2019 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 1–6.

[27] Sainathan Ganesh Iyer and Anurag Dipakumar Pawar. 2018. GPU and CPU accelerated mining of cryptocurrencies and their financial analysis. In *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on*. IEEE, 599–604.

[28] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*. Springer, 27–46.

[29] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: multilevel traffic classification in the dark. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. 229–240.

[30] Monero Research Lab. 2018. MONERO, a Private Digital Currency. https://www.getmonero.org/.

[31] Monero Research Lab. 2021. Monero Documentation - CryptoNight, a memory hard hash function. https://monerodocs.org/proof-of-work/cryptonight/.

[32] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. 2020. A survey on the security of blockchain systems. *Future Generation Computer Systems* 107 (2020), 841–853.

[33] Xiaojun Liu, Wenbo Wang, Dusit Niyato, Narisa Zhao, and Ping Wang. 2018. Evolutionary game for mining pool selection in blockchain networks. *IEEE Wireless Communications Letters* 7, 5 (2018), 760–763.

[34] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[35] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. 2006. *Wireshark & Ethereal network protocol analyzer toolkit.* Elsevier.

[36] Eva Papadogiannaki and Sotiris Ioannidis. 2021. A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[37] Antonio Pastor, Alberto Mozo, Stanislav Vakaruk, Daniele Canavese, Diego R López, Leonardo Regano, Sandra Gómez-Canaval, and Antonio Lioy. 2020. Detection of encrypted cryptomining malware connections with machine and deep learning. *IEEE Access* 8 (2020), 158036–158055.

[38] Ivan Petrov, Luca Invernizzi, and Elie Bursztein. 2020. Coinpolice: Detecting hidden cryptojacking attacks with neural networks. *arXiv preprint arXiv:2006.10861* (2020).

[39] Peter Phaal, Sonia Panchen, and Neil McKee. 2001. RFC3176: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks.

[40] Ruben Recabarren and Bogdan Carbunar. 2017. Hardening stratum, the bitcoin pool mining protocol. *Proceedings on Privacy Enhancing Technologies* 3 (2017), 57–74.

[41] Luigi Rizzo. 2012. netmap: a novel framework for fast packet I/O. In *21st USENIX Security Symposium (USENIX Security 12)*. 101–112.

[42] Rashid Tahir, Sultan Durrani, Faizan Ahmed, Hammas Saeed, Fareed Zaffar, and Saqib Ilyas. 2019. The browsers strike back: countering cryptojacking and parasitic miners on the web. In *IEEE Conference on Computer Communications*.

[43] Unit 42. 2021. Highlights from the Unit 42 Cloud Threat Report, 1H 2021. https://unit42.paloaltonetworks.com/highlights-cloud-threat-report-1h-2021/.

[44] Said Varlioglu, Bilal Gonen, Murat Ozer, and Mehmet F Bastug. 2020. Is Cryptojacking Dead after Coinhive Shutdown? *arXiv preprint arXiv:2001.02975* (2020).

[45] Tao Wang and Ian Goldberg. 2017. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1375–1390.

[46] Wenhao Wang, Benjamin Ferrell, Xiaoyang Xu, Kevin W Hamlen, and Shuang Hao. 2018. Seismic: Secure in-lined script monitors for interrupting cryptojacks. In *European Symposium on Research in Computer Security*. Springer, 122–142.

[47] Charles V Wright, Scott E Coull, and Fabian Monrose. 2009. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.. In *NDSS*, Vol. 9. Citeseer.

[48] Jun Xu, Jinliang Fan, Mostafa Ammar, and Sue B Moon. 2001. On the design and performance of prefix-preserving IP traffic trace anonymization. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. 263–266.

[49] ZeroDot1. 2020. Simple lists that can help prevent cryptomining in the browser or other applications. https://gitlab.com/ZeroDot1/CoinBlockerLists.

[50] Shize Zhang, Zhiliang Wang, Jiahai Yang, Xin Cheng, XiaoQian Ma, Hui Zhang, Bo Wang, Zimu Li, and Jianping Wu. 2021. MineHunter: A Practical Cryptomining Traffic Detection Algorithm Based on Time Series Tracking. In *Annual Computer Security Applications Conference*. 1051–1063.

[51] Aaron Zimba, Zhaoshun Wang, Mwenge Mulenga, and Nickson Herbert Odongo. 2018. Crypto mining attacks in information systems: An emerging threat to cyber security. *Journal of Computer Information Systems* (2018), 1–12.