Russo *et al. EURASIP Journal on Information Security* (2021) 2021:11
https://doi.org/10.1186/s13635-021-00126-1

EURASIP Journal on Information Security

RESEARCH Open Access

Detection of illicit cryptomining using network metadata

Michele Russo[1*], Nedim Šrndić[1] and Pavel Laskov[2]

Abstract

Illicit cryptocurrency mining has become one of the prevalent methods for monetization of computer security incidents. In this attack, victims' computing resources are abused to mine cryptocurrency for the benefit of attackers. The most popular illicitly mined digital coin is Monero as it provides strong anonymity and is efficiently mined on CPUs. Illicit mining crucially relies on communication between compromised systems and remote mining pools using the *de facto standard* protocol Stratum. While prior research primarily focused on endpoint-based detection of in-browser mining, in this paper, we address network-based detection of cryptomining malware in general. We propose XMR-Ray, a machine learning detector using novel features based on reconstructing the Stratum protocol from raw NetFlow records. Our detector is trained offline using *only mining traffic* and does not require privacy-sensitive normal network traffic, which facilitates its adoption and integration.

In our experiments, XMR-Ray attained 98.94% detection rate at 0.05% false alarm rate, outperforming the closest competitor. Our evaluation furthermore demonstrates that it reliably detects previously unseen mining pools, is robust against common obfuscation techniques such as encryption and proxies, and is applicable to mining in the browser or by compiled binaries. Finally, by deploying our detector in a large university network, we show its effectiveness in protecting real-world systems.

Keywords: Detection, Malware, Cryptomining, Monero, NetFlow, Machine learning, One-class classification

1 Introduction

At the end of 2017, the cryptocurrency market reached a market capitalization of over $600 billion [1]. Since then, the market has demonstrated its interest in this technology, and cryptocurrencies have proven to be a revolutionary asset class. In May 2021, another surge of the cryptocurrency market occurred, taking the market valuation to a record of over $2 trillion [2]. However, the potential financial gains attracted not only investors but also malicious actors.

Security risks related to cryptocurrencies are diverse. Conventional hacking incidents at cryptocurrency exchanges (e.g., [3, 4]) led to vast financial losses and even, as in the case of MtGox, to bankruptcy. Other prominent incidents included manipulation of wallet addresses [5], exploitation of wallet software [6–8], and compromise of mining power exchanges such as NiceHash [9]. In contrast to such jackpot-style incidents, illicit mining of digital coins [10] is a far more reliable source of income for criminals.

Various sources have reported a dramatic rise in cryptocurrency mining malware in the last 2 years. The year 2018 began with a surge in mining attacks [11], the trend rising in the entire year [12], during which time the number of cryptomining malware samples grew by more than 4000% [13]. In the first quarter of 2019, the number of campaigns targeting victims' computers to mine cryptocurrencies reportedly increased by 29% [14], and the recovery in trading price starting from May 2019 has resulted in a spike in cryptomining malware operations [15, 16]. Finally, large-scale attacks targeting enterprises are still widespread in 2020 (e.g., [17–19]).

*Correspondence: michele.russo1@huawei.com
[1]Huawei Technologies Duesseldorf GmbH, Munich, Germany
Full list of author information is available at the end of the article

Our work is focused on detecting mining of Monero cryptocurrency (abbr. *XMR*), the most prevalent one in the illicit cryptomining ecosystem due to its anonymity guarantees and the feasibility to use CPUs for mining [20]. Monero mining is typically performed by a group of miners (clients) connected to a common mining pool (server) which shares the workload and profit among miners. This arrangement boosts the chance for the entire pool to mine a new block and consequently obtain a reward for it, in turn maximizing the long-term expected gain for individual miners. The de facto standard application layer protocol for communication between pool servers and clients is *Stratum* [21]. It has a minimalist syntax and uses simple logic for pool mining, which triggered our interest to investigate if it can be discriminated from other network traffic.

Illicit cryptomining attacks can be categorized into two main families. *In-browser cryptomining* (also cryptojacking, browser-based mining) runs in the victim's web browser for as long as she stays on a web page with a mining script, e.g., CRYPTOLOOT [22]. *Binary-based cryptomining* malware is typically delivered via trojans which download and execute mining binaries as background processes. In both cases, by abusing computational power of hundreds of hijacked devices, attackers can amass significant computational power and generate substabtial earnings. Section 2 presents further details on "benign" and "malicious" mining.

From the victims' perspective, illicit cryptomining incurs electricity and cloud costs, and causes devices to slow down and deteriorate rapidly, to the point of potentially being physically destroyed [23]. Crucially, it should be recognized as a *breach indicator* and promptly rectified to prevent further serious damage, e.g., confidential data theft.

Due to these security implications, illicit cryptomining has received growing attention in the research community, cf. Section 3. The majority of previous work focused on understanding and assessing profitability of in-browser mining [24–27]; other work discussed its detection and mitigation [28–31]. Much less attention has been paid to binary-based cryptomining, comprising essentially the early investigation of Bitcoin mining botnets [32] and the comprehensive study of Monero mining malware [20]. In terms of detection techniques, the majority of prior work spotted mining on the endpoints by monitoring CPU or GPU usage and process metrics (processor time, system calls, number of threads, etc.), or in the network using simple indicators of compromise like domain and IP blacklists or IPS rules.

While endpoint-based approaches are appealing in their capability to identify salient features of cryptomining, their deployment is labor-intensive. Network-based detection, in contrast, can be localized on critical network nodes, but existing approaches cannot cope with proxies and encryption. In this work, we demonstrate that essential properties of cryptomining activity can be accurately recovered from *aggregated network traffic*, i.e., NetFlow records, thus combining the advantages of both approaches.

The unique capabilities of our proposed detector XMR-RAY stem from key design decisions elaborated in Section 5.1. It inspects network traffic metadata, i.e., NetFlow records (see example in Table 1). To alleviate the information loss in comparison to a full traffic dump, we introduce our principal innovation: a novel set of features based on constraint-solving which indicate whether a set of NetFlow records corresponding to a TCP session exhibits unique statistical regularities characteristic to Stratum. Using our novel features, Stratum traffic can be discriminated from other traffic by deploying a one-class classifier trained *solely on mining traffic* obtained from legitimate mining pool clients. As a result, we obtain a highly efficient, privacy-preserving detector resistant to encryption, tunneling and proxies, and outperforming detectors based on DPI. Its design and implementation are presented along with data collection in Section 5.

In Section 6, we show that XMR-RAY, *trained once* on legitimate cleartext mining traffic in our lab, successfully generalizes to a variety of use cases. First, we evaluate it in a controlled environment with traffic collected from a large corporate network (Section 5.3). Next, we assess its robustness against encryption as well as tunneling and apply it to both in-browser and binary-based mining traffic. We also show that it successfully detects traffic generated by mining malware. Finally, we deploy it in a large university network and assess its false alarm rate.

The main contributions of this work are:

- We design XMR-RAY, a machine learning system for detection of cryptocurrency mining.
- We propose a novel set of features for machine learning based on a form of constraint-solving which indicate whether NetFlow records corresponding to a

Table 1 NetFlow records corresponding to the traffic window depicted in Fig. 2

Source	Destination	Packets	Bytes	Start time	End time	Src. port	Dst. port	TCP flags	Protocol
Miner	Pool	11	1752	1530189072	1530189144	50122	7777	24	6
Pool	Miner	6	952	1530189072	1530189144	7777	50122	24	6

TCP stream exhibit unique statistical regularities characteristic to Stratum.
- We evaluate the detection performance of the proposed solution on a comprehensive collection of real-world traffic and compare it to the closest related prior work.
- We assess the system's robustness against encryption, tunneling, proxying, and adversarial machine learning.
- We assess the false-positive rate of our system in a real-world deployment in a large university.

Following the experimental evaluation, we discuss the main limitations of XMR-Ray in Section 7 and conclude the paper.

2 Cryptocurrency mining

Mining is a fundamental operation for cryptocurrency networks. Every transaction added to a public ledger must be verified to avoid over- or double-spending. In the absence of a central authority, participants of the peer-to-peer network verify several transactions of their choice. Transactions are grouped into blocks that are chained together by applying cryptographic hash functions to the content of the new block as well as the hash values of previous blocks, thus creating a tamper-proof "block-chain" in the ledger. Two mechanisms ensure that each participant verifies transactions honestly. First, obtaining the right to permanently add a block to the ledger is made expensive. Hence, every participant needs to "invest" resources into verification and provide an easily verifiable proof-of-work (PoW) for each block. Second, honest work is adequately rewarded.

Both mechanisms outlined above are implemented by mining which entails the contest for finding a padding to a block such that the hash function computed over the padded block evaluates to a value smaller than the given threshold. Cryptographic hash functions are irreversible; hence, the only way to solve this problem is to brute-force the padding. The first participant to find a suitable padding announces the solution to the network and receives a small value in cryptocurrency as a reward. The threshold, and hence the problem difficulty, is adjusted such that a solution is expected once in a certain time interval. The participants' chance of find a solution first is proportional to their share of computational resources within the network.

2.1 Mining pools

For miners without specialized computing resources, pooling together is the only realistic chance to gain any reward. A mining pool server, connected to a cryptocurrency network, distributes tasks to pool participants (clients). When a client finds a solution, it reports it to the server, which in turn verifies it and broadcasts it to the cryptocurrency network. Once the solution is verified by the network, the server receives the reward, withholds a commission and distributes the rest among clients according to their work.

A specialized application layer protocol called Stratum was developed for pooled mining. It is a line-based text protocol using TCP sockets and JSON-encoded messages. Although there is no official standard for Stratum and its implementations may differ for different currencies, its workflow remains largely the same across implementations. Mining pools typically implement the following operations using Stratum:

1. Broadcast mining jobs to the pool participants
2. Collect the participants' hashes/shares
3. Look for the block reward
4. Track the participants' contributions and distribute their rewards proportionally

Pooled mining is particularly popular for ASIC-resistant currencies like Monero, Dashcoin, and AEON, as they use a PoW algorithm designed to be egalitarian and efficient to compute on CPUs.

2.2 Malicious mining

Illegitimate or *malicious mining* refers to mining performed on hijacked resources. For example, employees may mine on a corporate computing cluster, or malware may use the CPU of an infected endpoint for monetization. There exist two prevalent techniques for malicious mining.

2.2.1 Binary-based cryptomining

This type of mining is performed by malware delivered via spam, exploit kits, or trojans. Once a device is infected, it downloads and starts a mining binary (executable). Previous studies reveal that cryptomining malware usually is a variant of a legitimate open-source mining client with custom configuration parameters, e.g., hard-coded address of the miscreant's cryptocurrency wallet, address of the mining pool or proxy, etc [20].

Monero (abbrev. XMR) is the preferred currency used by cryptomining malware, and XMRIG [33] is a popular legitimate mining tool deployed in most illicit mining campaigns [20, 34]. Since 2017, the number of coin miners has surged, with almost 4 million new samples in the third quarter of 2018 alone [13]. While the mining functionality has remained stable, its delivery methods and operational mechanisms have evolved with the malware ecosystem. Fileless malware [35] spreads using a combination of PowerShell and the EternalBlue exploit [36]. Some samples terminate rival mining malware on victim hosts to maximize their resource use [37]. Ransomware enhanced with

cryptocurrency mining capability decides in real time which of the two strategies is more lucrative [38].

2.2.2 In-browser mining

This type of mining takes place in users' web browsers while they are visiting a web page with an active mining script. It is considered legitimate if the site owner obtains the user's consent and may be financially more attractive to the owner than web advertisement [39]. In-browser Monero mining was pioneered by COINHIVE [40], which provided an efficient JavaScript mining client for embedding into websites, with claims of achieving up to 65% of the hashing rate of binary-based mining.

However, this type of mining is often abused by criminals who inject JavaScript mining scripts into vulnerable web sites. A number of such attacks has been reported using COINHIVE scripts on, e.g., government websites of the USA, the UK, and Australia [41]. Attackers also abused Google's DoubleClick service, making the affected web pages show legitimate advertisement while a web miner covertly performed mining in the background [42]. Another insidious attack vector targeted carrier-grade MikroTik routers vulnerable to remote login with authentication bypass [43]. In this case, attackers abused the routers' web proxy functionality to inject COINHIVE scripts into websites browsed by users who accessed the Web through them.

3 Related work

Below, we present and categorize the most relevant work addressing the phenomenon of cryptojacking. For a comprehensive and detailed review of prior work, we refer the reader to [44]. In addition, we place this work in the broader context of threat detection and traffic classification.

3.1 Detection of non-mining threats in network traffic

While detection initially focused on network packets [45, 46], the rising link speeds made packet analysis infeasible and alternative approaches have emerged based on network aggregated flows, especially NetFlow [47]. Combined flow- and packet-based approaches were introduced for malware family classification [48, 49]. Pure flow-based systems have been successful in detecting malware-related activities [50–52] like botnets [53–55], DoS attacks [56–58], scans [59–61], and worms [62, 63]. Similarly, our work is based on NetFlow, but we introduce novel features specialized for mining detection.

3.2 Traffic classification

The problem addressed in our work may be seen as a special case of traffic classification, namely, identification of the Stratum protocol traffic. Hence, our method is related to the respective work on traffic classification. Conti et al. proposed a method for identifying user actions in Android applications based on information from TCP/IP packet metadata in encrypted traffic [64, 65]. Papadogiannaki et al. proposed a pattern language for describing IP packet sequences and use it for fine-grained identification of application events, also in encrypted traffic [66]. Traffic classification was also performed using first 5 packets of a TCP connection [67] or counting received packets and bytes [68]. The impact of traffic sampling on performance of detection systems was studied in [69]. We refer the interested reader to [70, 71] for an exhaustive literature survey in traffic analysis and classification.

3.3 Longitudinal studies

A considerable body of related work investigated the cryptojacking ecosystem. In 2014, Huang et al. reverse-engineered a large population of Bitcoin mining malware and analyzed its network traffic to study operational behavior, geographical distribution, and other features [32]. They identified coherent botnet campaigns, estimated their earnings and speculated about their payout strategies. A similar study has addressed the clandestine ecosystem of malicious Monero mining binaries [20]. Other longitudinal studies addressed in-browser cryptojacking, most prominently MINESWEEPER [28], presenting a new detection technique using static JavaScript analysis and monitoring CPU cache events during WebAssembly execution. An analysis of Alexa's top 1 million sites provided insights into campaigns' earnings estimates, distribution networks, pools, proxies, and services. Bijmans et al. [27] extend this work with a larger list of domains gathered from different sources. They analyze NetFlow records associated with mining services and assess their usage distribution. Other studies addressed similar questions, differing primarily in techniques for the detection of cryptojacking websites [24–26, 72–75].

3.4 Endpoint- and cloud-based detection

Early work on detection addressed Bitcoin mining in cloud environments. Solanas et al. [76] proposed a method for detecting undesirable activities, including Bitcoin mining, by using privacy-friendly features extracted from OpenStack implementations. They use metrics such as CPU utilization, disk read/write request rates/bytes and network byte/packet rates to train a detector. A similar approach was taken in [77]. Analogously, hardware-assisted profiling has been used to create discernible signatures for various mining algorithm [78, 79]. Other related work relies on analysis of power consumption, network logs and web resources [80], hardware performance indicators [81, 82], or on a combination of both Windows performance counters and network flows features [83]. RAPID analyzes endpoint performance from the

web browser to detect in-browser mining using machine learning in real time [29]. Naseem et al. convert Wasm binaries to gray-scale images and utilize a convolutional neural network-based classifier to label an image as either malicious (i.e., cryptojacking) or benign [84]. Cryptojacking was also detected by tracing opcodes during execution although not from the browser itself [31]. In the field of memory forensics, Ali et al. [85] proposed several techniques for extraction of key cryptojacking indicators, e.g., wallet addresses and Stratum protocol messages, from memory images. Side channels have also been successfully used for detection [86].

Detection of cryptojacking at the endpoints or in cloud infrastructure may appear straightforward since cryptojacking inherently incurs high CPU and memory usage. Nevertheless, it remains susceptible to false positives (processes with similar resource usage profiles) and evasion (attackers throttle the consumption to stay under the radar). Especially in enterprise environments, deployment and orchestration of endpoint detection may be very costly.

3.5 Network-based detection

Guidelines published in technical papers, e.g., [87], typically present simple network detection rules. Others, e.g., [88], list indicators of compromise (IoC) for Monero mining, e.g., pool domains, wallet addresses, C&C communication patterns. However, such techniques are routinely evaded by attackers. Academic research developed complementary ideas. Swedan et al. present a system for gateway-based traffic analysis which dissects HTTP(S) traffic to extract and analyze JavaScript in real time using heuristic rules [89]. This approach is nevertheless susceptible to evasion using JavaScript obfuscation and bears a substantial operational burden associated with HTTPS proxies. Several papers [90–93] rely on computing features upon packet flows and training binary classification machine learning models. They achieve high detection accuracy at the expenses of computation and deployment overhead.

Motivated to achieve accurate detection and reduce cost, recent papers investigate using NetFlow [94, 95]. They propose binary classification to discriminate between mining and non-mining traffic, utilizing the C4.5 decision tree algorithm and generic NetFlow-based features known from earlier work [96]. However, binary classification complicates deployment, as the model needs to be trained from scratch in every environment ("train everywhere"). In contrast, our detector is trained once using only mining traffic and can subsequently be deployed in any environment without retraining because it does not depend on the highly diverse non-mining traffic ("train once, deploy everywhere"). Furthermore, our work introduces novel features highly specialized for mining detection, and we experimentally demonstrate its superior detection performance.

4 Stratum traffic analysis

In this section, we present the details of the Stratum pool mining protocol. To this end, we perform a manual analysis of full packet traces of Stratum traffic captured in our lab and show the most important findings. Our traffic collection effort is documented in Section 5.3.

Communication over Stratum starts with the pool client (miner) contacting the pool server and performing authentication. As soon as the miner successfully logs into a mining pool, it starts receiving jobs over *New Job* messages which have the general format shown in Listing 1.

```
{
  "jsonrpc": "2.0",
  "method": "job",
  "params": {
    "blob": "152 hex char string (76 bytes)",
    "job_id": "16864",
    "target": "cf8b0000"
  }
}
```

Listing 1 Stratum new job message

In the examples, we show the format used by a specific combination of client-server tools, i.e., the mining pool `mine.xmrpool.net` and mining client XMRIG 2.6.4, but our algorithm is designed to be independent of the specific implementation and focuses on the Stratum workflow.

The main properties of *New Job* messages areas follows:

- `blob`: an ASCII hexadecimal string (76 bytes/152 characters) representing the content to be hashed by miners in order to produce Monero coins. A portion of this string is editable and represents the `nonce` that can be arbitrarily set by miners.
- `job_id`: a string used to match jobs with their corresponding results; it has a variable length which depends on the pool implementation.
- `target`: in order for the block to be accepted by the pool, its header hash must be lower than or equal to the current target, thus lowering the target makes the hash computation more difficult.

When a mining pool client receives a *New Job* message from the server, it starts the mining process. This involves finding a `nonce` such that the value generated by hashing the new `blob`, using the PoW hash function, is lower than the target. The information included in the *New Job* message is sufficient to define the problem.

Once the miner finds a solution, it submits its result to the pool in the form of a *Solution Submission* message. An example for this type of message can be seen in Listing 2. It contains several fields, but 3 are relevant for our analysis. The `job_id` corresponds to the job which has been

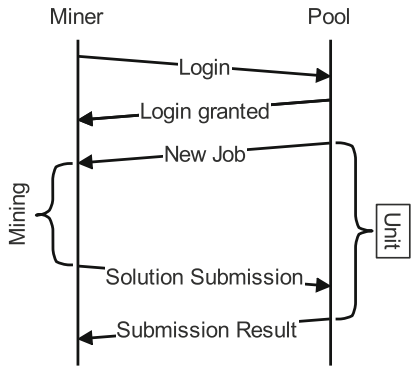

Fig. 1 Typical stratum mining workflow

completed. The `nonce` (8 hexadecimal characters denoting a 4-byte value) has been found by the miner and used to produce a suitable hash. The hash value of the block header using the found nonce is returned as `result`.

```
{
  "id": 76,
  "jsonrpc": "2.0",
  "method": "submit",
  "params": {
    "id": "1",
    "job_id": "16871",
    "nonce": "b51100e0",
    "result": "64 hex char string (32 bytes)"
  }
}
```

Listing 2 Stratum solution submission message

The mining pool server receives the *Solution Submission* message. After verifying the correctness of the embedded solution, it sends back a *Submission Result* message. An example is shown in Listing 3. This pattern of communication between the pool client and server, denoted in Fig. 1 as *Unit*, repeats until the network connection is terminated.

```
{
  "id": 76,
  "jsonrpc": "2.0",
  "result": {
    "status": "OK"
  },
  "error": null
}
```

Listing 3 Stratum Submission Result message.

A detailed example of the communication between a miner and a Monero public pool over Stratum is shown in Fig. 2. The two subfigures show the TCP/Stratum packets sent by the pool and the miner, respectively, during the mining process. The plot is based on full-packet capture (DPI) for demonstration purposes, but our algorithm works on NetFlow data. As an example, NetFlow records corresponding to the window of network traffic between the dashed vertical lines are presented in Table 1.

Miners may sometimes receive *New Job* messages from the pool while they are already working on a job. This happens for two reasons: either a new block has been mined or new transactions appeared for the current block. Therefore, the ratio between the number of jobs and number of the submitted solutions in a single NetFlow record pair is not one-to-one. This makes reconstructing Stratum semantics from NetFlow more difficult.

By analyzing several mining traces using visualization tools, e.g., see Fig. 2, we discovered the following properties:

- The *size* of Stratum messages exchanged between a server and client remains nearly constant.
- *New Job* and *Solution Submission* are the largest messages sent by the pool and the miner, respectively.
- After *Login*, the miner generally sends 2 types of messages: *Solution Submission* and *Keep-Alive*.
- The number of TCP acknowledgments (ACK) sent by the miner, excluding keep-alives, is equal to the number of *Submission Result* and *New Job* messages sent by the pool. The number of ACKs sent by the pool is not equal to the number of *Solution*

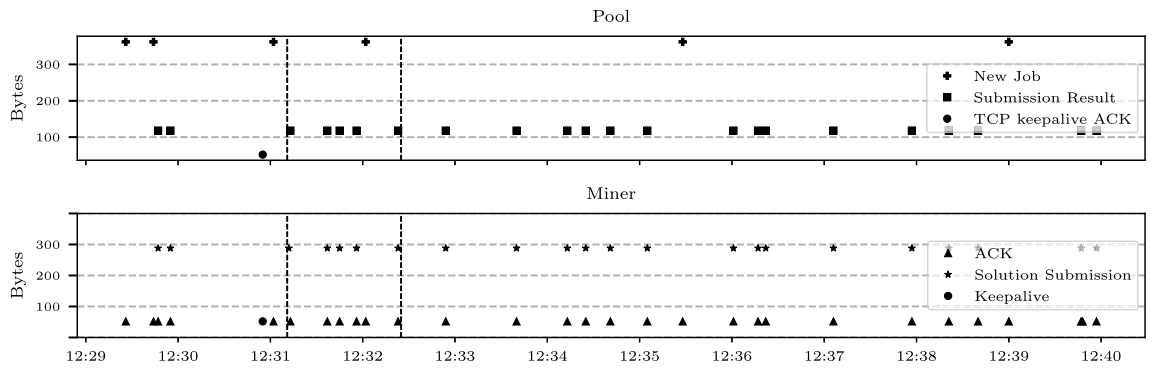

Fig. 2 Example of stratum communication between a mining pool and client

Submission. This happens because the pool often piggybacks the ACK flag for a *Solution Submission* in the *Submission Result* packet.

These properties enable us to engineer Stratum-specific features for accurate detection of mining traffic.

5 XMR-Ray detector

This section presents the design and implementation of the cryptomining detector XMR-RAY, with a particular focus on design decisions and its novel features.

5.1 Design tradeoffs

Several features of the current cryptomining ecosystem are crucial for the design of our detector. First, the Stratum protocol is de facto standard. Furthermore, not only the protocol but also the few client implementations are largely shared among legitimate and malicious users. Finally, malicious mining operations are often carried out via legitimate public mining pools [20]; thus, the malicious actors are forced to use exactly the same client and protocol implementations as the said public pools.

To take advantage of these findings, we made the following key design choices for XMR-RAY:

1. It is based on network traffic inspection, not endpoint monitoring,
2. It uses aggregate information, i.e., NetFlow, not DPI.
3. It employs one-class, not binary classification.
4. Its features are mining-specific.

Tradeoffs of these decisions are summarized in Table 2, empty cells denote cases with no discernible effects.

Installing and managing endpoint monitoring tools with comprehensive coverage is challenging in dynamic environments. Network-based detection, in contrast, can be centralized. However, existing simple network-based approaches, e.g., IP/domain blacklists and IPS rules, cannot cope with encryption. Consequently, XMR-RAY inspects NetFlow instead of packets, achieving several advantages while potentially reducing accuracy due to the information loss.

The choice of machine learning approach is based on the premise that pool mining traffic can be discriminated from all other network communications. Therefore, we employ a one-class classifier (OCC) trained solely on mining traffic generated by generic legitimate mining pool clients. This is a clear competitive advantage over related work which employs binary classifiers, as they additionally require full network traffic for training. In essence, binary classifiers decide between two classes, while OCCs decide between the target class and everything else, i.e., "universe". If the target class is well characterized by representative training data and characteristic features, and the universe is very diverse and constantly changing—an assumption which is true in our case—then OCC is clearly a superior choice. The benefits are very significant, the only drawback is the information loss which may lead to lower accuracy.

To alleviate information loss, the main innovation effort was invested in reconstructing as much information as

Table 2 Summary of benefits and drawbacks of XMR-RAY's design

Design choice	Efficiency	Privacy	Cost-efficiency	Ease of deployment	Robustness	Accuracy
Network- vs. endpoint-based	Does not slow down endpoints.		Does not require endpoint agents.	Deployed and managed centrally.	Robust against endpoint evasion. Not applicable to novel protocols§.	
NetFlow vs. DPI	Several orders of magnitude less input to process.	Traffic content is not inspected.	Network devices, e.g., switches, already support NetFlow aggregation.	No need to decrypt traffic using an HTTPS proxy.	Relatively robust against encryption.	Information loss due to aggregation§.
One-class vs. binary classification	Amount of mining traffic used for OCC training is a small fraction of normal enterprise traffic necessary for training binary classifiers.	No need to use sensitive enterprise traffic for training, only generic Stratum traffic.	Minimal training data maintenance: collection of generic Stratum traffic is sufficient.	The classifier, as trained by security vendors, can be shipped to customers without on-site adaptation. Minimal retraining required.	Not affected by any changes in enterprise traffic.	Information loss due to lack of access to normal traffic§.
Specialized vs. generic NetFlow features	Computationally slightly more expensive§.				Robust against encryption, proxying, and tunneling.	Tailored to Stratum protocol semantics.

Sentences marked with "§" denote drawbacks

possible from NetFlow records by designing Stratum-specific features. Section 6 demonstrates that this effort was very successful and that another crucial benefit was achieved this way: robustness against encryption, proxying, and tunneling. The cost is a minor increase of computation effort, already compensated by using NetFlow and OCC.

Clearly, the benefits are numerous and very significant, especially for the last 2 design choices, which also represent competitive advantages compared to related work. However, the main strength is also the main weakness: by being Stratum-specific, the approach is not applicable to completely novel protocols. This is discussed from the perspective of adversarial machine learning in Section 6.6 and as a limitation in Section 7.

5.2 Deployment scenario

A typical deployment scenario for our detector is shown in Fig. 3, where XMR-RAY receives metadata of traffic passing through the network gateway. A network flow is a communication session between two applications described by the tuple (A_s, p_s, A_d, p_d, P), where A_s, A_d are source and destination IP addresses, p_s, p_d are the corresponding ports, and P is the IP protocol. Each direction in the communication is considered a separate, unidirectional flow.

NetFlow records are exported according to different timeout rules:

- *Active timeout* takes place upon expiration of the active timeout, enabling periodic export of flow statistics in long network conversations.
- *Inactive timeout* occurs if no packet is observed in a flow within a specified time interval.
- *Flag-based timeout* is activated when FIN and RST flags in TCP sessions indicate session termination.

After NetFlow collection is enabled on a device, flow statistics are stored and updated in a cache. When a flow times out, its statistics are exported in form of a NetFlow record.

5.3 Data collection

The starting point of our mining investigation is a representative corpus of legitimate mining traffic collected using a mining server optimized for CPU and GPU mining. Such a traffic is largely machine-independent and thus the machine learning models' results are not biased even if the data was generated and collected from a single machine. It runs Ubuntu 18.04 and has an AMD RX 580 GPU and AMD Ryzen 7 2700x CPU. The primary mining tools were XMRIG for CPU and CLAYMORE/XMR-STAK for GPU mining, as these are well maintained and widely used both legitimately and in cryptomining malware campaigns [20]. Traffic was collected by running TCPDUMP locally on the mining server and converted to NetFlow setting both active and inactive timeouts to 100s. In our implementation, metadata is encoded using NetFlow v5, the most common and lightweight version, but alternative formats like IPFIX are also suitable. This procedure is used for training and evaluation in our experiments.

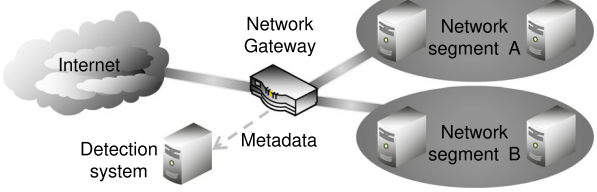

Fig. 3 Typical deployment scenario for XMR-RAY

To collect a comprehensive corpus of *cleartext* mining traffic, we mined in 25 well-known Monero public pools [97] for about 6 months, gathering around 4000 h of mining traffic.

For the experimental evaluation, we also collected a corpus of NetFlow data (test dataset) from a large enterprise network (about 10k hosts). Like for mining traffic, export timeouts were set to 100s. It comprises around 500 million flows, of which there are 16,000 TCP conversations (565,000 flows) longer than 30 min and was collected for 1 month. This heterogeneous network environment is representative for large enterprises. We assume that the test data does not contain mining traffic and our manual investigation did not find any, even in our method's false positives. Finally, it is worth noting that all NetFlow records were collected with a sampling rate of 100%. This is standard practice among enterprises that for security purposes deploy dedicated NetFlow exporters. To enable the computation of performance metrics like true/-false positive rate in our experiments, we inject subsets of mining conversations taken from our mining traffic collection into the test dataset. Specifically, we insert NetFlow records belonging to mining TCP conversations among the NetFlow records from the benign enterprise traffic. We carefully avoid using the injected mining traffic for training.

5.4 System architecture

The architecture of the XMR-RAY system is depicted in Fig. 4. It takes as input NetFlow records and operates in 2 modes: training and deployment. In training mode, NetFlow records from our database of collected mining traffic are used to train a machine learning model. In deployment mode, the trained model is used to classify NetFlow records resulting from TCP conversations of new, live network traffic as either mining or other traffic. In the following, we provide a detailed description of individual modules.

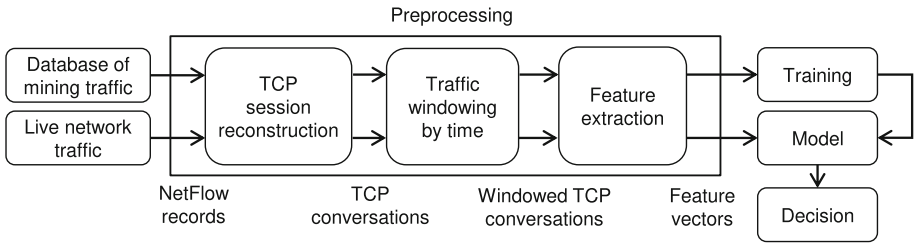

Fig. 4 System architecture

5.4.1 TCP conversation reconstruction

Stratum runs over TCP, and mining is carried out during long communication sessions which we denote as "conversations." NetFlow records are collected with a timeout that can be much shorter than the duration of a single conversation, e.g., 100 s. A single mining session therefore corresponds to many NetFlow records. The first step of our method is to reassemble unidirectional NetFlow records into bidirectional TCP conversations. All NetFlow records sharing the same TCP 4-tuple are grouped into a single TCP conversation.

5.4.2 Traffic windowing by time

In the second step, the previously reassembled TCP conversations are partitioned by time into overlapping successive windows, each window having the same time length of l_W seconds. This is repeated every f_W seconds. The overlap is due to the sliding window nature of splitting and $l_W \geq f_W$. Keeping l_W constant ensures that time-dependent features have a comparable value among all time windows. The sets of time-windowed TCP conversations produced in this step are the basic unit of processing for the machine learning model, i.e., the model's decisions are made based on NetFlow records collected within a time window of l_W seconds from a single TCP conversation.

5.4.3 Feature extraction

In this step, the output of the previous module is prepared for processing by the machine learning algorithm. The feature extraction module computes 110 features capable of capturing the intrinsic network behavior of Stratum communications. The features are described in Section 5.5. The output are real-valued *feature vectors* that describe the salient properties of individual traffic windows.

Feature extraction is the last module in the preprocessing subsystem. The preprocessing is performed in exactly the same way for both training and deployment.

5.4.4 Training

In the training mode, incoming feature vectors are used to train a one-class classification model. In this mode, *only mining traffic* from the training dataset is used as input. The model is trained to identify whether a set of NetFlow records corresponding to a time window of l_W seconds from a single TCP conversation contains mining traffic or not. The output of the training step is a one-class classification model.

5.4.5 Prediction

In prediction mode, the model predicts whether windows of live traffic represent mining or not. During deployment, decisions can also be made on a coarser granularity of TCP conversations by applying some rules (e.g., majority voting) to the outcome of window-based prediction. In all but the real-world deployment experiment (see Section 6.8), we use window-based prediction to evaluate our method. This is intended to evaluate the worst-case performance in the absence of error correction mechanisms.

5.5 Feature engineering

A crucial contribution of this work is a novel set of features that describe characteristic traits of the Stratum protocol from the information aggregated in NetFlow records. To this end, we devised a novel feature extraction process dubbed "speculative reconstruction." The analysis of Stratum traffic that motivated our feature design is presented in Section 4. All features are computed for NetFlow records corresponding to individual traffic windows of duration l_W seconds of TCP conversations.

5.5.1 Group 1: Heuristics based on speculative reconstruction of Stratum

The first group of features exploits the fact that Stratum messages have a very constrained format, which enables us to reconstruct characteristics of cleartext messages from aggregated values in NetFlow records. We dub this process "speculative" because the constraints we have identified do not always have a unique solution but nevertheless enable a good guess about the true sequence of Stratum messages.

The process starts by analyzing each pair of corresponding unidirectional NetFlow records in a time-windowed TCP conversation, i.e., the NetFlow record from A to B

and B to A. Our goal is to estimate, for each pair of records, the following quintuples:

1. ACK packet size,
2. Size of *New Job* messages
3. Size of *Solution Submission* messages
4. Number of *New Job* messages
5. Number of *Solution Submission* messages

To this end, we apply 10 intrinsic constraints imposed by TCP and Stratum, for example:

- ACK packet size in bytes $\in \{40, 44, \ldots, 80\}$.
- Size of *Solution Submission* messages is greater than the size of ACK packets by at least 72 bytes (nonce + result).
- Size of *New Job* messages is greater than the size of ACK packets by at least 152 bytes (blob size).
- Number of miner's packets is the sum of number of ACK packets, *Keep-Alive* and *Solution Submission* messages.
- Number of miner's ACK packets is the sum of the numbers of *New Job* and *Submission Result* messages.

After filtering out *Keep-Alive* messages, we guess the ACK packet size and estimate the remaining four quantities so as to match the number of packets and bytes observed in NetFlow records. After discarding quintuples with non-integer values we are usually left with several plausible quintuples, i.e., solutions, for each pair of NetFlow records.

In the second step, we group all quintuples obtained for a time-windowed TCP conversation by the first 3 dimensions (i.e., ACK packet size, size of *New Job* messages, size of *Solution Submission* messages) and count their occurrences. These 3 dimensions are imposed by TCP and Stratum configuration parameters of the pool and miner and they remain nearly constant for the duration of the session. The last 2 dimensions (i.e., number of *New Job* messages and number of *Solution Submission* messages) are variables reflecting the current workload of the miner.

In the final step, we use the triplet counts from the second step to compute the following features for every window of l_W seconds:

F1.1 Ratio of the largest triplet count to the total count of feasible triplets
F1.2 Difference between largest and smallest triplet count
F1.3 Standard deviation of triplet counts
F1.4 Ratio of *New Job* packet size to *Solution Submission* packet size for the most frequent triplet
F1.5 Standard deviation of *New Job* message counts across all NetFlow records for the most frequent triplet
F1.6 Standard deviation of *Solution Submission* message counts across all NetFlow records for the most frequent triplet.

The feature sensitivity analysis of our classifier showed that these features play a major role in attaining high detection accuracy. In the following, we attempt to elucidate the speculative reconstruction on an example with an l_W of 30 min. We use the same value for the experiments in Section 6.

The essence of the speculative reconstruction process is solving a system of equations whose constraints are dictated primarily by Stratum semantics. As a result, in case of a mining conversation, a single solution of the system should frequently repeat across the conversation's NetFlow record pairs. On the other hand, no solution should dominate others in non-mining conversations.

Features **F1.1–F1.3** explicitly aim at capturing this characteristic. A high value for these features means that only one triplet (ACK packet size, size of *New Job* messages, size of *Solution Submission* messages) solves the equation system for most NetFlow record pairs. This is a strong indicator of mining traffic. Figure 5 shows the statistical distribution of features **F1.1–F1.3** for approximately 15,000 mining and normal conversations of l_W minutes. Although features computed on mining conversations show some variability in their values, their medians are several times higher than those of normal conversations. This demonstrates the validity of speculative reconstruction and the importance of features **F1.1–F1.3** as indicators of mining traffic.

Features **F1.4–F1.6** characterize the average difficulty of jobs requested by specific pools. For non-mining traffic, they are likely to appear random. In addition, **F1.6** captures the effects of the VARDIFF algorithm [98] employed by pools to update the miners' job difficulty based on their average time to finish past jobs.

While speculative reconstruction features carry a strong signal for differentiating between mining and other traf-

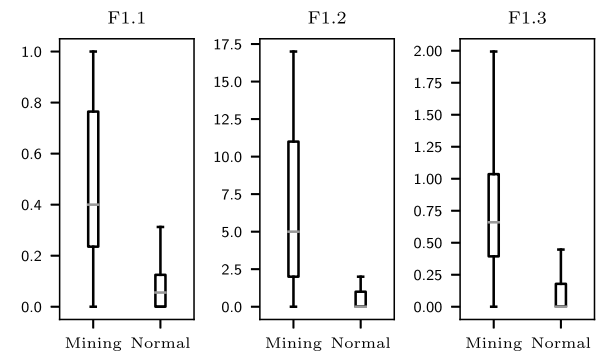

Fig. 5 Statistical distribution of F1.1, F1.2, and F1.3

fic, alone they cannot achieve low false positive rate. We observed during feature development that the most similar traffic to Stratum was VoIP and protocols exhibiting polling or beaconing behaviors. These protocols share commonalities with Stratum, such as a near-constant transmission rate, long duration, and repeating messages. To finally distinguish between mining and these similar protocols, we extended our feature set with the following.

5.5.2 Group 2: Correlation between mean packet sizes of the miner and the pool

The ratio of byte count and packet count of a NetFlow record gives its mean packet size. For the mean packet sizes of 2 sequences of unidirectional NetFlow records of every time window, we compute different correlation metrics, e.g., Pearson correlation coefficient. This captures the intuition that a *New Job* message from the pool is unlikely to be immediately followed by a solution from the miner. In NetFlow terms, the pool's record has a high mean packet size (since *New Job* is the biggest Stratum message), and the corresponding miner's record a low one, because it acknowledges the *New Job* but does not deliver the solution in the same record.

In the other direction, when the miner sends a *Solution Submission* it is unlikely that the pool has sent a *New Job* message in the previous seconds. Also, the pool needs to send a *Submission Result* and occasionally an ACK, which lowers the mean packet size since these packets are much smaller than *New Job* messages. Hence the two lists of mean packet sizes are anti-correlated for mining traffic and uncorrelated for non-mining traffic.

5.5.3 Group 3: Correlation between packet count and mean packet size

Features in this group are similar to group 2 but instead correlate the mean packet size and packet count of all NetFlow records with the same direction. We observe that a miner's *Solution Submission* triggers a response from the pool that the miner needs to ACK. Hence, a *Solution Submission* message boosts the number of packets sent by both sides, causing the mean packet size of the miner to raise (*Solution Submission* is miner's biggest packet) and of the pool to drop (*Submission Result* is small). Hence, the miner's packet count and mean packet size are correlated, the opposite is true for the pool, and no correlation should be observed for most other traffic.

5.5.4 Group 4: Ratio of transmitted bytes

For each unidirectional NetFlow record (A to B), we find the corresponding answer (B to A) and compute the ratio of bytes sent and received. The features represent the distribution of such ratios: range, interquartile range, mean, and standard deviation.

5.5.5 Group 5: Mean packet size

These features represent the distribution of mean packet sizes for NetFlow records with the same direction, comprising standard deviation, range, and interquartile range.

5.5.6 Group 6: Transmission rate

These features represent the distribution of the transmission rate (bytes/s) for NetFlow records with the same direction, comprising standard deviation, interquartile range and minimum rate.

Feature groups 4–6 capture the intuition that mining communication has a finite set of possible packets exchanged, as well as relatively constant miner-pool communication, following a specific pattern. Finally, the above semantically motivated features are complemented with useful general features derived from number of sent bytes and mean packet inter-arrival time.

5.6 One-class classification

In machine learning, one-class classification (OCC) attempts to identify objects of a specific class by learning from a training set containing only objects of that class. The task in OCC is to define a classification boundary around the positive (target) class, such that it accepts as many target objects as possible, while minimizing the chance of accepting negative (outlier) objects. It is a one-vs-rest classification, where the rest are not observed during training.

XMR-RAY uses the *Isolation Forest* OCC algorithm as implemented in SCIKIT-LEARN [99]. Its hyperparameters are tuned using grid search; the following values are chosen for all subsequent experiments: $n\_estimator = 300$, $contamination = 0.01$, *max_samples* is set to the number of training samples. At an early stage in research, we evaluated other one-class classifiers, such as one-class SVM, but their results were inferior. Given a generally high accuracy of XMR-RAY, we did not study even more advanced classifiers, e.g., deep learning.

6 Results

In this section, we test XMR-RAY in diverse environments, with the goal of providing a comprehensive and realistic evaluation.

For the detection in networks with thousands of hosts it is crucial to maximize detection rate (i.e., recall or true positive rate (TPR)) and minimize false alarm rate (i.e., false positive rate (FPR)):

$$\mathrm{TPR} = \frac{TP}{P}, \qquad \mathrm{FPR} = \frac{FP}{N}.$$

Above, TP represents the count of true alarms, P the total count of mining traffic windows, FP the count of false alarms, and N the total count of non-mining windows.

A single pair of (TPR, FPR) values provides a very limited view of detector's performance. Most detectors have variable detection thresholds (e.g., the percentage of trees in a random forest which vote for one of the classes) which, when adjusted, change their decisions and therefore (TPR, FPR) rates as well. The receiver operating characteristic (ROC) curve provides a comprehensive view of detector's performance because it incorporates all (TPR, FPR) operating points of the detector for all values of its detection threshold (as a 2D curve). Throughout this section, we evaluate the detection performance using the area under ROC (AuROC), which provides a summary of the ROC curve. Its values lie in the range (0, 1), the higher the better.

6.1 Trade-off between latency and accuracy

Our first experiment aims to find a good trade-off between detection latency and accuracy, as influenced by the window size l_W, a parameter of our algorithm. Lowering its value reduces latency but also the amount of information available to make the decision.

For each window size l_W, we perform a variant of a 10-fold cross-validation. Our set of mining traffic comprises traffic collected from all 25 recorded mining pools (see Section 5.3). We divide the mining traffic into 10 different 80:20 splits. The 80% comprises the training set. The remaining 20% is mixed with our non-mining enterprise traffic collection (also described in Section 5.3) to serve as the test set.

Figure 6 shows that the detection accuracy grows with l_W, as expected. We choose l_W = 1800s (30min), with TPR = (98.94 ± 0.34)% and FPR = (0.054 ± 0.027)%, and use this value in all subsequent experiments. The FPR is computed per time window (30min) of a TCP stream, rather than per flow record. In our test environment with about 3500 TCP conversations per day which exceed 30 min, XMR-RAY produced about 3 false positives per day.

6.2 Detection of novel mining pools

Another crucial property of cryptomining detectors is their capability to detect traffic from miners that communicate with previously unknown mining pools. This is a challenging task since mining traffic can be, and in practice already is, easily customized. Indeed, the majority of well-known Monero public pools use existing customizable tools like NODE-CRYPTONOTE-POOL [100] and NODEJS-POOL [101].

To evaluate XMR-RAY's ability to detect novel pools, we design a new experiment. Similarly to the previous case, we use 10-fold cross-validation, but this time, we make sure that none of the pools in the test set have appeared in the training set. This experiment simulates real-world deployment where completely new pools appear regularly. We evaluate several ratios of numbers of pools used for training and testing, ranging from 7:18 to 22:3. Experiment results depicted in Fig. 7 show a clear advantage in using more pools for training and give a promise that model performance has potential to improve with time as more mining traffic is collected for training. Using about 10 pools for training suffices to achieve high TPR (96.20 ± 3.04)% on the remaining 15 previously unseen pools with an FPR of only (0.058 ± 0.071)%.

6.3 Detection of malware mining traffic

To verify that XMR-RAY detects mining by malware, we set up a CUCKOO sandbox [102] and executed two samples belonging to mining campaigns WEBCOBRA [103] and NRSMINER [104]. We collected around 30 h of traffic thus generating an insignificant profit for the threat actors. We noticed that the WEBCOBRA miner connected to an IP address, while NRSMINER used a DNS query. In both cases, the remote hosts may have been proxies or private pool servers, as they did not belong to any public Monero pools.

Nevertheless, XMR-RAY correctly classified all 30-min (l_W) time windows as mining. This experiment does not evaluate FPR by design, as the model is only applied to positive test samples. The result corroborates our hypothesis that there is no substantial difference between legitimate and illegitimate mining traffic.

6.4 Detection in obfuscated network traffic

Another experiment evaluates XMR-RAY's capability to correctly identify obfuscated mining traffic, i.e., using

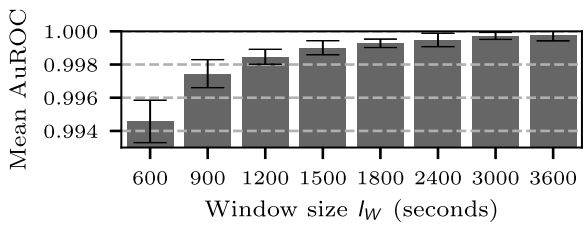

Fig. 6 Detection performance as a function of window size l_W

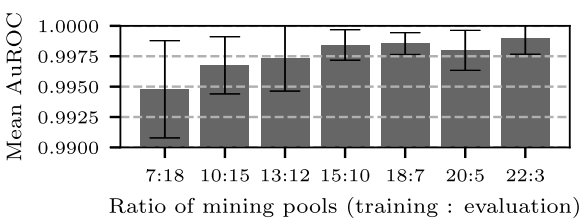

Fig. 7 Detection performance on novel mining pools

encryption or connecting through a tunnel, even when it was *trained only on NetFlow records of cleartext mining traffic*. Similarly to the previous experiment, in this section, we do not evaluate FPR by design, as the model is only applied to positive test samples.

6.4.1 Encrypted mining

Since the beginning of 2018, mining pools and tools increasingly provide the option to perform mining over an encrypted channel using SSL/TLS. In some cases, pools encourage the use of SSL by reducing fees for miners that do so [105]. In October 2018, XMRIG introduced support for SSL/TLS.

To collect encrypted mining traffic, we mined for several hours with different mining tools in 11 out of 25 pools that supported SSL mining. All pools accepted one of the following ciphers proposed by the client: `AES{128,256}-GCM-SHA{256,384}` [106]. This choice is natural since `AES-GCM` has slowly replaced `AES-CBC` over the past years, becoming the most used encryption mechanism. Furthermore, TLS1.3 no longer supports CBC-mode ciphers since `AES-GCM` combines stronger security, lower traffic overhead and higher performance on modern hardware.

Counter mode of operation is designed to turn block ciphers into stream ciphers; hence, we expected `AES-GCM` (Galois Counter Mode) to increase the packet size but keep the crucial properties and relations we observed in Stratum messages. And indeed, the model detected 98.16% of the 1000 evaluated 30-min windows of SSL mining traffic. Although this finding is not surprising given our feature design, it is nevertheless a remarkable accomplishment for a machine learning algorithm trained without access to encrypted traffic.

In the next step, we modified the mining client to propose other cipher suites in the TLS handshake. The only remaining ones the pools accepted were `AES{128,256}-CBC-SHA{128,256}`. The detection rate was 99.35% on 250 h of encrypted mining traffic, thus the padding of the CBC mode of operation has a negligible influence on our features. From these results, we conclude that our model reliably detects encrypted mining traffic, due to two main reasons:

1. Reliance on metadata instead of payload
2. Custom feature design that reveals the intrinsic behavioral characteristics of the Stratum protocol.

We are not aware of any use of SSL/TLS by cryptomining malware so far. However, encryption is becoming ever more widespread and it is not difficult to imagine its use for this purpose in the near future. By proposing a model resistant to evasion using encryption, we hope to be one step ahead of malicious actors.

6.4.2 SOCKS proxy

This is an SSH tunnel which applications use to forward their traffic to a proxy server that relays it to the final destination. The clients must use an SSH agent.

For this experiment, we collected several hours of mining traffic through a SOCKS 5 proxy using ciphers `CHACHA20-POLY1305`, `AES{128,192,256}-CTR`, and `AES{128,256}-GCM`. To test a different cipher suite from the previous experiment, we force the SSH session to use the AEAD cipher `CHACHA20-POLY1305`. On 250 h of mining over an SSH tunnel, the detection rate was 89.16%.

6.4.3 Mining proxy

In the past few years, several botnets started mining, and the mining pools started to ban them under the pressure of the mining community. A common approach for spotting botnets is to count the number of IP addresses that connect using the same wallet address (e.g., `mineXMR.com` [107]). To avoid a ban and mask their wallets, attackers started using mining proxies like XMRIG-PROXY [108]. Such tools are similar to standard proxies but communicate over Stratum and manage a large number of miners. They reduce the number of connections to the pool, e.g., up to 256 times for XMRIG-PROXY. XMR-Ray has detected 90.08% of the 200 h of mining that we performed over XMRIG-PROXY.

6.5 Detection of in-browser mining

Previous longitudinal studies investigated the distribution of well-known cryptomining services in web pages [25, 27, 28]. Our collection included the top 3: COINHIVE, COINIMP [109], and CRYPTOLOOT [22]. These were found in around 75% of all websites with mining scripts within the Alexa Top 1 million list, COINHIVE alone accounted for 60%. We found the websites hosting these scripts via the search engine PublicWWW [110], connected to them to perform mining and collected the traffic. The scripts throttled the CPU usage down to 30% to evade detection.

Table 3 shows that XMR-RAY reliably detects COINHIVE and CRYPTOLOOT. Similarly to the previous two experiments, we do not evaluate its FPR by design, as the model is only applied to positive test samples. We further investigate the reasons for a somewhat lower detection rate for COINIMP.

Table 3 Detection results for in-browser mining

Mining service	Traffic (h)	Detection rate
Coinhive	250	98.85%
CryptoLoot	100	95.26%
CoinImp	75	79.47%

During in-browser mining, clients communicate with the mining pool using WebSockets. All our collected examples used WebSockets over TLS. By collecting and decrypting a new session, we noticed CoinIMP also used obfuscation. Looking at the distribution of packet sizes in the packet dump (before conversion to NetFlow), we noticed that CoinIMP introduced variations to the Stratum workflow that reduced our detection rate. We thus conclude that the WebSocket proxy server alters the client-side protocol.

On March 8, 2019, Coinhive stopped its mining services. As a result, other mining services took advantage of Coinhive's absence and cryptojacking attacks are still widespread [111–113].

6.6 Robustness against adversarial evasion

Machine learning algorithms are vulnerable to adversarial examples, i.e., data which is manipulated in order to evade a specific detector. We implement several software modifications for the XMRig mining client as a preliminary evaluation of XMR-Ray's robustness against adversarial evasion. Our modifications are intended to trick our model into misclassifying mining TCP conversations as non-mining. Similarly to previous experiments, we do not evaluate FPR by design, as the model is only applied to positive test samples. Results of our 3 evasions are shown in Table 4.

6.6.1 Attack 1: Data injection

Analyzing the open-source code of mining pools, we noticed that the JSON parser function extracts only the key-value pairs it is interested in and discards the others. We then add to each Stratum *Solution Submission* message a new key-value pair of random length between 32 and 512 bytes. This modification had little influence on the detection rate.

6.6.2 Attack 2: Message injection

If the JSON key *method* is missing from a client message, the pool does not send a Stratum reply but simply a TCP ACK. We make XMRig send random-length messages at random intervals every 2–10s. This attack considerably reduces the detection rate. Modifying the number of packets and randomizing packet sizes disrupts our client-side features.

6.6.3 Attack 3: Error triggering

The last example is the one that, among the three, alters the mining traffic shape and behavior the most. We modified the mining client to send messages with a wrong value for the *method* JSON key. We noticed that certain public pools answer with an error message that contains the same ID of the received faulty message. Therefore, by sending messages with a wrong method and variable-length ID at random intervals, we can trigger variable-length error messages in response from the pool. Doing so, we manage to evade the detection around 50% of the time. This does not come as a surprise since a modification like this heavily affects the standard Stratum workflow and thus both our client- and pool-side features. This technique works only with pools that deploy the standard NODE-CRYPTONOTE-POOL.

From our experiments, we conclude XMR-Ray can indeed be vulnerable to evasion attacks. However, for the attacks to work, the adversary must deviate from the standard Stratum protocol. Nevertheless, such deviations from the protocol can be both detected and sanctioned by security updates to pool software.

Table 4 Detection results for adversarial evasion

Attack	Traffic (h)	Detection rate
Data injection	250	93.37%
Message injection	350	81.28%
Error triggering	200	53.16%

6.7 Comparison to prior work

To position our system among existing detectors, we compare it to its most closely related prior work, Muñoz et al. [94]. This work also follows a network-based approach using NetFlow but, in contrast to XMR-Ray, employing generic features and binary classification. We reimplemented it to the best of our knowledge and describe the technical details of our reimplementation in Appendix A.

Figure 8 compares the detection performance of both approaches on detecting novel mining pools, as in Section 6.2. The results for XMR-Ray, repeated from Fig. 7, are clearly superb. In particular, using 10 training pools it achieved TPR = (96.20 ± 3.04)% and FPR = (0.058 ± 0.071)%, while the competitor had *about 28x* higher FPR = (1.65 ± 1.07)% at a lower detection rate of (94.96 ± 2.60)%.

The detection rate of the approaches was compared across all experiments described in Sections 6.3-6.6, as summarized in Table 5. Malware is fully detected by both. Muñoz et al. is more robust against stream ciphers but a lot more vulnerable against CBC-mode encryption. Other experiments, most notably adversarial evasion, provide rich empirical evidence for the superiority of XMR-Ray. Given the boost of TPR and reduction of FPR, we

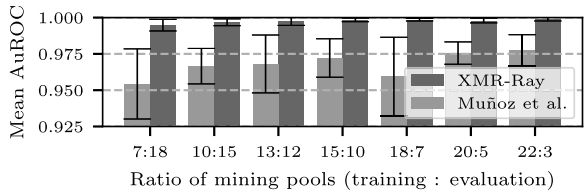

Fig. 8 Comparison on detecting novel mining pools

Table 5 Competitive comparison

		Detection rate (%)	
Category	**Experiment**	**Muñoz et al.**	**XMR-Ray**
Malware	NRSminer	**100.00**	**100.00**
	WebCobra	**100.00**	**100.00**
Encryption	SSL stream	**99.13**	98.16
	SSL CBC	90.11	**99.35**
Proxying	SOCKS5	88.35	**89.16**
	XMRIG	82.92	**90.08**
In-browser mining	Coinhive	90.82	**98.85**
	CryptoLoot	80.80	**95.26**
	CoinIMP	68.52	**79.47**
Adversarial evasion	Data injection	66.08	**93.37**
	Msg. injection	25.96	**81.28**
	Err. triggering	16.89	**53.16**

conclude that XMR-RAY's specialized features and OCC approach provide a substantial improvement over the state of the art.

6.8 Results from real-world deployment

In the final stage of our research we deployed XMR-RAY with the trained model described in Section 6.1 in the network of a large university with around 100,000 endpoints. As a common procedure for practical deployments, we used an error-correction mechanism for the model's predictions which we refer to as majority voting. Our model was run every 3 hours, loading the past 3 hours of traffic, and labeling a TCP conversation as mining only if the majority of its traffic windows have been labeled as such. We configured the traffic windowing module to use an overlap of $f_W = l_W/2 = 900s$, thus having at most 11 traffic windows within 3 h. To assist security analysts in incident prioritization, we categorized each TCP conversation (not limited to 3 h) into low or high risk, depending on the count and ratio of its windows classified as mining.

In 2 weeks of operation in the university network, XMR-RAY analyzed 2,219,410 TCP conversations longer than $l_W = 1800$ s and labeled 52 as mining: 5 with high and 47 with low risk. We manually investigated the positives to assess the detection performance. The 5 high-risk conversations were confirmed as Monero mining. Interestingly, only one host was mining by connecting directly to a well-known public pool (pool.supportxmr.com:7777), while the others were mining through proxies. Two of these proxies' IP addresses were associated with malware activity according to AbuseIPDB [114]. In addition, one was listed as IoC for a Chinese cryptomining hacking campaign targeting macOS users [115]. We were not able to confirm any of the 47 low-risk positives; thus, the system achieved a FPR of 0.0021%, i.e., 3.4 false positives per day. The risk categorization has helped the analysts focus their attention on legitimate threats.

7 Discussion

The experimental evaluation presented above demonstrates that the proposed method has clearly met the objectives of its design. In this section, we elucidate design limitations and discuss possible solutions for overcoming them, as well as further applications of our design ideas for NetFlow traffic analysis.

One limitation of our design is its lack of support for UDP. No existing cryptomining implementations known to us use UDP, but an attacker could, in principle, tunnel Stratum over UDP. To close such evasion opportunity, unidirectional NetFlow features could be investigated that do not rely on any TCP communication patterns.

Another feature of our design is its limitation to Stratum. If pools migrate to novel protocols, our method will become ineffective. However, we find this scenario to be unlikely for the near future. Stratum is widely deployed, proven, and reliable for its users, and we are not aware of plans to replace it. In case of cryptojackers, to avoid using Stratum, they would have to develop both a custom mining client and the pool which, on top of being costly, represents a single point of failure and brings maintenance overhead due to Monero hard forks. Although Stratum will be replaced at some point in the future, the same is true for all protocols, including HTTP and TLS, which went through major changes recently. And just like new detectors were necessary for HTTP/2 and TLS 1.3, researchers will need to make new detectors for future pool mining protocols, an evolutionary approach typical for security.

Adversarial examples are a further threat to our technique, just like to all other machine learning approaches. As shown in Table 4, some hand-crafted obfuscations indeed reduce the detection accuracy. However, for the protocol deviations to succeed, they must be accepted by the pool. Legitimate public pools are incentivized to and do block deviating miners and we expect that our proposed attacks will be rendered ineffective as soon as they become deployed. Illegitimate pools are free to change the protocol, in which case the evasion attack is reduced to protocol modification attack which we addressed in the previous paragraph.

An exciting direction for future work is deeper investigation of the speculative reconstruction technique. In the feature sensitivity analysis for our classifier (omitted due to lack of space) we have observed that the features obtained by this technique (group 1) play the major role in attaining high detection accuracy. It is there-

fore interesting to understand what other applications of network metadata analysis can benefit from this design technique but also whether it may negatively affect user privacy. The interconnection of various design decisions enabled by this technique requires a better understanding of its mathematical properties, especially the existence and uniqueness of solutions for different protocols.

8 Conclusions

In this paper, we addressed the problem of detecting illegitimate cryptocurrency mining in network traffic. We demonstrated that the de facto standard protocol for pooled mining, Stratum, exhibits salient patterns which enable its reliable detection using NetFlow. The presented system XMR-RAY outperforms the closest competitors by combining one-class classification (OCC) with novel Stratum-specific features.

OCC is trained solely on mining traffic generated by legitimate mining clients, while binary classifiers employed by the state of the art also require full non-mining network traffic. In case the target class is well characterized by representative training data and characteristic features, and the other classes are very diverse and constantly changing, an assumption which holds true for cryptocurrency mining, then OCC is a superior choice and provides major benefits in efficiency, privacy, costs, deployment and robustness.

Our principal innovation is a set of features using constraint-solving to assess whether NetFlow records corresponding to a TCP stream are likely to originate from Stratum traffic. These features also provide another crucial benefit: robustness against encryption, proxying and tunneling.

A comprehensive experimental evaluation in a large corporate environment demonstrated high effectiveness. With a detection latency of 30 min, XMR-RAY reached a detection accuracy of 98.94% at a false alarm rate of 0.05%, about 28 times lower than that of the closest competitor. We have also demonstrated its ability to detect traffic towards novel mining pools, as well as to most prominent cryptojacking services. Deployed in a large university network it successfully detected real-world mining attacks.

Akin to other machine learning systems, our technique is not immune to adversarial examples. We discuss this and other limitations of our approach, and outline the prospects for using our novel feature extraction technique in other applications of network metadata analysis.

Appendix A

To compare our detector to the state of the art, we reimplemented the detector described in [94]. The features are as follows:

- Inbound and outbound packets/second
- Inbound and outbound bytes/second
- Inbound and outbound bytes/packet
- Bytes_inbound/bytes_outbound ratio
- Packets_inbound/packets_outbound ratio

We are confident about the quality of our reimplementation since, even if not fully described, the baseline features' names are self-explanatory and leave little room for interpretation.

For a fair and accurate comparison, the goal was to evaluate both approaches following the same procedure. However, that is generally not possible because they are trained on different data: OCC only on mining while binary classifiers also require non-mining traffic. Therefore, for training binary classifiers, we resorted to another corpus of NetFlow data from the same enterprise environment where the evaluation was performed. To simulate a realistic deployment, the non-mining part of the binary classifiers' training traffic was collected a month before the test dataset described in Section 5.3. Crucially, the entire test dataset and the mining portion of the training dataset remains identical.

We repeated the experiments of Sections 6.1-6.6 for 2 best models from [94]: C4.5 and CART. We used WEKA, specifically the PYTHON-WEKA-WRAPPER library [116], to implement the C4.5 algorithm and SCIKIT-LEARN for CART. In our environment, CART performed significantly better, and therefore, we compare only its results against ours.

Abbreviations

AES: Advanced encryption standard; ASCII: American standard code for information interchange; ASIC: Application-specific integrated circuit; AuROC: Area under receiver operating characteristic; CART: Classification and regression trees; CBC: Cipher block chaining; CPU: Central processing unit; CTR: Counter; DPI: Deep packet inspection; FPR: False-positive rate; GCM: Galois/counter mode; GPU: Graphics processing unit; HTTP: Hypertext transfer protocol; HTTPS: Hypertext transfer protocol secure; IoC: Indicator of compromise; IP: Internet protocol; IPFIX: Internet protocol flow information export; IPS: Intrusion prevention system; JSON: JavaScript object notation; OCC: One-class classification; PoW: Proof of work; ROC: Receiver operating characteristic; SHA: Secure hash algorithm; SSL: Secure sockets layer; TCP: Transport control protocol; TLS: Transport layer security; TPR: True-positive rate; UDP: User datagram protocol; XMR: Monero cryptocurrency

Acknowledgements

Not applicable.

Authors' contributions

MR performed the data acquisition and software implementation and the main part of data analysis and interpretation. NŠ participated in data analysis and interpretation. All authors participated in the conception, design, and manuscript writing. All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Due to the commercially and privacy-sensitive nature of the research, no data subjects consented to their data being retained or shared.

Declarations

Competing interests

The authors are currently applying for patents relating to the content of the manuscript. MR and NŠ have received salary from an organization that has applied for patents relating to the content of the manuscript.

Author details

[1]Huawei Technologies Duesseldorf GmbH, Munich, Germany. [2]University of Liechtenstein, Vaduz, Liechtenstein.

Received: 15 June 2021 Accepted: 10 November 2021
Published online: 04 December 2021

References

1. S. Higgins, $600 billion: cryptocurrency market cap sets new record (2017). https://www.coindesk.com/600-billion-cryptocurrency-market-cap-sets-new-record/ Accessed 19 Apr 2021
2. CoinMarketCap, Global cryptocurrency charts - total market capitalization (2021). https://coinmarketcap.com/charts/ Accessed 17 May 2021
3. M. Yamazaki, Tokyo-based cryptocurrency exchange hacked, losing $530 million: NHK (2018). https://www.reuters.com/article/us-japan-cryptocurrency/tokyo-based-cryptocurrency-exchange-hacked-losing-530-million-nhk-idUSKBN1FF29C Accessed 21 Apr 2021
4. J. Russell, Korean crypto exchange Bithumb says it lost over $30M following a hack (2018). https://techcrunch.com/2018/06/19/korean-crypto-exchange-bithumb-says-it-lost-over-30m-following-a-hack/ Accessed 21 Apr 2021
5. M. Yuval, CoinDash TGE Hack findings report 15.11.17 (2017). https://blog.coindash.io/coindash-tge-hack-findings-report-15-11-17-9657465192e1 Accessed 21 Apr 2021
6. E. Ananin, A. Semenchenko, Copy-pasting thief from a copy-pasted code (2018). https://www.fortinet.com/blog/threat-research/copy-pasting-thief-from-a-copy-pasted-code.html Accessed 21 Apr 2021
7. J. Wilmoth, Hackers seize $32 million in Ethereum in parity wallet breach (2017). https://www.ccn.com/hackers-seize-32-million-in-parity-wallet-breach/ Accessed 21 Apr 2021
8. S. Higgins, Tether claims $30 million in US dollar token stolen (2017). https://www.coindesk.com/tether-claims-30-million-stable-token-stolen-attacker Accessed 21 Apr 2021
9. R. Iyengar, More than $70 million stolen in bitcoin hack (2017). https://money.cnn.com/2017/12/07/technology/nicehash-bitcoin-theft-hacking/index.html Accessed 21 Apr 2021
10. C. Budd, Threat brief: malware authors mine Monero across the globe in a big way (2108). https://unit42.paloaltonetworks.com/threat-brief-malware-authors-mine-monero-across-globe-big-way/ Accessed 21 Apr 2021
11. E. Lopatin, Kaspersky Security Bulletin 2018 story of the year: miners (2018). https://securelist.com/kaspersky-security-bulletin-2018-story-of-the-year-miners/89096/ Accessed 21 Apr 2021
12. European Union Agency For Network and Information Security, Enisa threat landscape report 2018. Technical report, ENISA (2019). https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018
13. McAfee Labs, McAfee Labs Threats Report (2018). https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2018.pdf Accessed 21 Apr 2021
14. McAfee Labs, McAfee Labs Threats Report (2019). https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf Accessed 21 Apr 2021
15. C. Cimpanu, Cryptomining malware saw new life over the summer as Monero value tripled (2019). https://www.zdnet.com/article/crypto-mining-malware-saw-new-life-over-the-summer-as-monero-value-tripled/ Accessed 21 Apr 2021
16. AMR, Kaspersky Security Bulletin 2019 (2019). https://securelist.com/kaspersky-security-bulletin-2019-statistics/95475 Accessed 21 Apr 2021
17. C. Cimpanu, Thousands of enterprise systems infected by new Blue Mockingbird malware gang (2020). https://www.zdnet.com/article/thousands-of-enterprise-systems-infected-by-new-blue-mockingbird-malware-gang/ Accessed 21 Apr 2021
18. J. Karasek, A. Remillano, Outlaw updates kit to kill older miner versions, targets more systems (2020). https://blog.trendmicro.com/trendlabs-security-intelligence/outlaw-updates-kit-to-kill-older-miner-versions-targets-more-systems/ Accessed 21 Apr 2021
19. A. Windsor, Breaking down a two-year run of Vivin's cryptominers (2020). https://blog.talosintelligence.com/2020/01/vivin-cryptomining-campaigns.html Accessed 21 Apr 2021
20. S. Pastrana, G. Suarez-Tangil, in *Proceedings of the Internet Measurement Conference, IMC*. A first look at the crypto-mining malware ecosystem: a decade of unrestricted wealth (ACM, Amsterdam, 2019), pp. 73–86. https://doi.org/10.1145/3355369.3355576
21. BitcoinWiki, Stratum mining protocol (2020). https://en.bitcoinwiki.org/wiki/Stratum_mining_protocol Accessed 21 Apr 2021
22. Crypto-Loot, CryptoLoot earn more from your visitors (2020). https://crypto-loot.org/ Accessed 21 Apr 2021
23. N. Buchka, A. Kivva, D. Galov, Jack of all trades (2017). https://securelist.com/jack-of-all-trades/83470/ Accessed 21 Apr 2021
24. S. Eskandari, A. Leoutsarakos, T. Mursch, J. Clark, in *IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops*. A first look at browser-based cryptojacking (IEEE, London, 2018), pp. 58–66. https://doi.org/10.1109/EuroSPW.2018.00014
25. J. Rüth, T. Zimmermann, K. Wolsing, O. Hohlfeld, in *Proceedings of the Internet Measurement Conference IMC*. Digging into browser-based crypto mining (ACM, Boston, 2018), pp. 70–76. https://dl.acm.org/citation.cfm?id=3278539
26. G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, M. Yang, Y. Zhang, Z. Qian, H. Duan, in *ACM SIGSAC Conference on Computer and Communications Security CCS*, ed. by D. Lie, M. Mannan, M. Backes, and X. Wang. How you get shot in the back: a systematical study about cryptojacking in the real world (ACM, Toronto, 2018), pp. 1701–1713. https://doi.org/10.1145/3243734.3243840
27. H. L. J. Bijmans, T. M. Booij, C. Doerr, in *28th USENIX Security Symposium, USENIX Security*, ed. by N. Heninger, P. Traynor. Inadvertently making cyber criminals rich: a comprehensive study of cryptojacking campaigns at internet scale (USENIX Association, Santa Clara, 2019), pp. 1627–1644. https://www.usenix.org/conference/usenixsecurity19/presentation/bijmans
28. R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, G. Vigna, in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, ed. by D. Lie, M. Mannan, M. Backes, and X. Wang. MineSweeper: an in-depth look into drive-by cryptocurrency mining and its defense (ACM, Toronto, 2018), pp. 1714–1730. https://doi.org/10.1145/3243734.3243858
29. J. D. P. Rodriguez, J. Posegga, in *Proceedings of the 34th Annual Computer Security Applications Conference ACSAC*. RAPID: resource and API-based detection against in-browser miners (ACM, San Juan, 2018), pp. 313–326. https://doi.org/10.1145/3274694.3274735
30. W. Wang, B. Ferrell, X. Xu, K. W. Hamlen, S. Hao, in *23rd European Symposium on Research in Computer Security*, ed. by J. López, J. Zhou, and M. Soriano. SEISMIC: SEcure In-lined Script Monitors for Interrupting Cryptojacks, vol. 11099 (Springer, Barcelona, 2018), pp. 122–142. https://doi.org/10.1007/978-3-319-98989-1_7
31. D. Carlin, P. O'Kane, S. Sezer, J. Burgess, in *16th Annual Conference on Privacy, Security and Trust*, ed. by K. McLaughlin, A. A. Ghorbani, S. Sezer, R. Lu, L. Chen, R. H. Deng, P. Miller, S. Marsh, and J. R. C. Nurse. Detecting cryptomining using dynamic analysis (IEEE Computer Society, Belfast, 2018), pp. 1–6. https://doi.org/10.1109/PST.2018.8514167
32. D. Y. Huang, H. Dharmdasani, S. Meiklejohn, V. Dave, C. Grier, D. McCoy, S. Savage, N. Weaver, A. C. Snoeren, K. Levchenko, in *21st Annual Network and Distributed System Security Symposium*. Botcoin: monetizing stolen cycles (The Internet Society, San Diego, 2014). https://www.ndss-symposium.org/ndss2014/botcoin-monetizing-stolen-cycles
33. xmrig, XMRig (2021). https://xmrig.com/ Accessed 21 Apr 2021
34. J. Grunzweig, The rise of the cryptocurrency miners (2018). https://unit42.paloaltonetworks.com/unit42-rise-cryptocurrency-miners/ Accessed 21 Apr 2021
35. V. Bulavas, A. Kazantsev, A mining multitool (2018). https://securelist.com/a-mining-multitool/86950/ Accessed 21 Apr 2021
36. SecurityFocus, Microsoft Windows SMB Server CVE-2017-0144 Remote Code Execution Vulnerability (2017). https://www.securityfocus.com/bid/96704 Accessed 21 Apr 2021
37. Dr. WEB Anti-virus, Linux.BtcMine.174 (2018). https://vms.drweb.com/virus/?i=17645163 Accessed 21 Apr 2021

38. E. Vasilenko, O. Mamedov, To crypt, or to mine - that is the question (2018). https://securelist.com/to-crypt-or-to-mine-that-is-the-question/86307/ Accessed 21 Apr 2021
39. P. Papadopoulos, P. Ilia, E. P. Markatos, Truth in web mining: Measuring the profitability and cost of cryptominers as a web monetization model. CoRR (2018). http://arxiv.org/abs/1806.01994
40. Coinhive, Coinhive Monero JavaScript Mining (2019). https://web.archive.org/web/20190109010215/https://coinhive.com/ Accessed 21 Apr 2021
41. C. Leonard, CoinHive cryptocurrency mining script injected into 1000s of government websites via BrowseAloud plugin (2018). https://www.forcepoint.com/blog/x-labs/coinhive-cryptocurrency-mining-script-injected-1000s-government-websites Accessed 21 Apr 2021
42. C. Liu, J. Chen, Google's DoubleClick abused to deliver miners (2018). https://blog.trendmicro.com/trendlabs-security-intelligence/malvertising-campaign-abuses-googles-doubleclick-to-deliver-cryptocurrency-miners/ Accessed 21 Apr 2021
43. M. Hron, D. Jursa, MikroTik mayhem: Cryptomining campaign abusing routers (2018). https://blog.avast.com/mikrotik-routers-targeted-by-cryptomining-campaign-avast Accessed 21 Apr 2021
44. E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, A. A. Selcuk, SoK: cryptojacking malware (2021). https://doi.org/2103.03851
45. R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, S. Zhou, in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ed. by V. Atluri. Specification-based anomaly detection: a new approach for detecting network intrusions (ACM, Washington, 2002), pp. 265–274. https://doi.org/10.1145/586110.586146
46. Y. Huang, W. Lee, in *7th International Workshop on Recent Advances in Intrusion Detection*, ed. by E. Jonsson, A. Valdes, and M. Almgren. Attack analysis and detection for ad hoc routing protocols, vol. 3224 (Springer, Sophia Antipolis, 2004), pp. 125–145. https://doi.org/10.1007/978-3-540-30143-1_7
47. F. Erlacher, F. Dressler, On high-speed flow-based intrusion detection using snort-compatible signatures. IEEE Trans Dependable Secure Comput., 1–1 (2020)
48. B. Anderson, D. A. McGrew, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity (ACM, Halifax, 2017), pp. 1723–1732. https://doi.org/10.1145/3097983.3098163
49. X. Deng, J. Mirkovic, in *11th USENIX Workshop on Cyber Security Experimentation and Test*, ed. by C. S. Collberg, P. A. H. Peterson. Malware analysis through high-level behavior (USENIX Association, Baltimore, 2018). https://www.usenix.org/conference/cset18/presentation/deng
50. K. Bartos, M. Sofka, V. Franc, in *25th USENIX Security Symposium, USENIX Security*, ed. by T. Holz, S. Savage. Optimized invariant representation of network traffic for detecting unseen malware variants (USENIX Association, Austin, 2016), pp. 807–822. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bartos
51. B. A. Alahmadi, I. Martinovic, in *2018 APWG Symposium on Electronic Crime Research, eCrime*. MalClassifier: malware family classification using network flow sequence behaviour (IEEE, San Diego, 2018), pp. 1–13. https://doi.org/10.1109/ECRIME.2018.8376209
52. M. Piskozub, R. Spolaor, I. Martinovic, Malalert: detecting malware in large-scale network traffic using statistical features. SIGMETRICS Perform. Evaluation Rev. 46(3), 151–154 (2018). https://doi.org/10.1145/3308897.3308961
53. G. Gu, R. Perdisci, J. Zhang, W. Lee, in *Proceedings of the 17th USENIX Security Symposium*, ed. by P. C. van Oorschot. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection (USENIX Association, San Jose, 2008), pp. 139–154. http://www.usenix.org/events/sec08/tech/full_papers/gu/gu.pdf
54. L. Bilge, D. Balzarotti, W. K. Robertson, E. Kirda, C. Kruegel, ed. by R. H. Zakon. 28th Annual Computer Security Applications Conference (ACM, Orlando, 2012), pp. 129–138. https://doi.org/10.1145/2420950.2420969
55. D. Zhao, I. Traoré, B. Sayed, W. Lu, S. Saad, A. A. Ghorbani, D. Garant, Botnet detection based on traffic behavior analysis and flow intervals. Comput. Secur. 39, 2–16 (2013). https://doi.org/10.1016/j.cose.2013.04.007
56. Y. Gao, Z. Li, Y. Chen, in *26th IEEE International Conference on Distributed Computing Systems (ICDCS)*. A DoS resilient flow-level intrusion detection approach for high-speed networks (IEEE Computer Society, Lisboa, 2006), p. 39. https://doi.org/10.1109/ICDCS.2006.6
57. A. Lakhina, M. Crovella, C. Diot, in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ed. by R. Guérin, R. Govindan, and G. Minshall. Mining anomalies using traffic feature distributions (ACM, Philadelphia, 2005), pp. 217–228. https://doi.org/10.1145/1080091.1080118
58. G. Münz, G. Carle, in *10th IFIP/IEEE International Symposium on Integrated Network Management*. Real-time analysis of flow data for network attack detection (IEEE, Munich, 2007), pp. 100–108. https://doi.org/10.1109/INM.2007.374774
59. A. Sperotto, R. Sadre, A. Pras, in *IP Operations and Management, 8th IEEE International Workshop*, ed. by N. Akar, M. Pióro, and C. Skianis. Anomaly characterization in flow-based traffic time series, vol. 5275 (Springer, Samos Island, 2008), pp. 15–27. https://doi.org/10.1007/978-3-540-87357-0_2
60. Q. Zhao, J. J. Xu, A. Kumar, Detection of super sources and destinations in high-speed networks: algorithms, analysis and evaluation. IEEE J. Sel. Areas Commun. 24(10), 1840–1852 (2006). https://doi.org/10.1109/JSAC.2006.877139
61. I. Paredes-Oliva, P. Barlet-Ros, J. Solé-Pareta, in *International Workshop on Traffic Monitoring and Analysis*, ed. by M. Papadopouli, P. Owezarski, and A. Pras. Portscan detection with sampled NetFlow, vol. 5537 (Springer, Aachen, 2009), pp. 26–33. https://doi.org/10.1007/978-3-642-01645-5_4
62. T. Dübendorfer, B. Plattner, in *14th IEEE International Workshops on Enabling Technologies (WETICE)*. Host behaviour based early detection of worm outbreaks in internet backbones (IEEE Computer Society, Linköping, 2005), pp. 166–171. https://doi.org/10.1109/WETICE.2005.40
63. T. Dübendorfer, A. Wagner, B. Plattner, in *Proceedings of the First IEEE International Workshop on Critical Infrastructure Protection IWCIP '05*. A framework for real-time worm attack detection and backbone monitoring (IEEE Computer Society, USA, 2005), pp. 3–12. https://doi.org/10.1109/IWCIP.2005.2
64. M. Conti, L. V. Mancini, R. Spolaor, N. V. Verde, Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Inf. Forensics Secur. 11(1), 114–125 (2016). https://doi.org/10.1109/TIFS.2015.2478741
65. M. Conti, L. V. Mancini, R. Spolaor, N. V. Verde, in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, CODASPY*, ed. by J. Park, A. C. Squicciarini. Can't you hear me knocking: identification of user actions on Android apps via traffic analysis (ACM, San Antonio, 2015), pp. 297–304. https://doi.org/10.1145/2699026.2699119
66. E. Papadogiannaki, C. Halevidis, P. Akritidis, L. Koromilas, ed. by M. Bailey, T. Holz, M. Stamatogiannakis, and S. Ioannidis. 21st International Symposium on Research in Attacks, Intrusions, and Defenses, vol. 11050 (Springer, Heraklion, 2018), pp. 315–334. https://doi.org/10.1007/978-3-030-00470-5_15
67. L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, K. Salamatian, Traffic classification on the fly. Comput. Commun. Rev. 36(2), 23–26 (2006). https://doi.org/10.1145/1129582.1129589
68. D. Rossi, S. Valenti, in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, ed. by A. Helmy, P. Mueller, and Y. Zhang. Fine-grained traffic classification with NetFlow data (ACM, Caen, 2010), pp. 479–483. https://doi.org/10.1145/1815396.1815507
69. V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, J. Solé-Pareta, Analysis of the impact of sampling on netflow traffic classification. Comput. Netw. 55(5), 1083–1099 (2011). https://doi.org/10.1016/j.comnet.2010.11.002
70. T. T. T. Nguyen, G. J. Armitage, A survey of techniques for internet traffic classification using machine learning. IEEE Commun. Surv. Tutor. 10(1-4), 56–76 (2008). https://doi.org/10.1109/SURV.2008.080406
71. M. Conti, Q. Li, A. Maragno, R. Spolaor, The dark side(-channel) of mobile devices: a survey on network traffic analysis. IEEE Commun. Surv. Tutor. 20(4), 2658–2713 (2018). https://doi.org/10.1109/COMST.2018.2843533
72. J. Rauchberger, S. Schrittwieser, T. Dam, R. Luh, D. Buhov, G. Pötzelsberger, H. Kim, in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ed. by S. Doerr, M. Fischer, S. Schrittwieser, and D. Herrmann. The other side of the coin: a framework for detecting and analyzing web-based cryptocurrency mining campaigns (ACM, Hamburg, 2018), pp. 18–11810. https://doi.org/10.1145/3230833.3230869
73. M. Musch, C. Wressnegger, M. Johns, K. Rieck, Web-based cryptojacking in the wild. CoRR abs/1808.09474 (2018). http://arxiv.org/abs/1808.09474

74. I. Petrov, L. Invernizzi, E. Bursztein, CoinPolice: detecting hidden cryptojacking attacks with neural networks (2020). http://arxiv.org/abs/2006.10861
75. A. Romano, Y. Zheng, W. Wang, in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Minerray: semantics-aware analysis for ever-evolving cryptojacking detection, (2020), pp. 1129–1140
76. M. Solanas, J. Hernandez-Castro, D. Dutta, Detecting fraudulent activity in a cloud using privacy-friendly data aggregates. CoRR abs/1411.6721 (2014). http://arxiv.org/abs/1411.6721
77. R. Ning, C. Wang, C. Xin, J. Li, L. Zhu, H. Wu, in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Capjack: capture in-browser crypto-jacking by deep capsule network through behavioral analysis, (2019), pp. 1873–1881. https://doi.org/10.1109/INFOCOM.2019.8737381
78. R. Tahir, M. Huzaifa, A. Das, M. Ahmad, C. A. Gunter, F. Zaffar, M. Caesar, N. Borisov, in *20th International Symposium on Research in Attacks, Intrusions, and Defenses*, ed. by M. Dacier, M. Bailey, M. Polychronakis, and M. Antonakakis. Mining on someone else's dime: mitigating covert mining operations in clouds and enterprises, vol. 10453 (Springer, Atlanta, 2017), pp. 287–310. https://doi.org/10.1007/978-3-319-66332-6_13
79. M. Conti, A. Gangwal, G. Lain, S. G. Piazzetta, Detecting covert cryptomining using HPC. CoRR abs/1909.00268 (2019). http://arxiv.org/abs/1909.00268
80. M. Bissaliyev, A. Nyussupov, S. Mussiraliyeva, Enterprise security assessment framework for cryptocurrency mining based on monero. J. Math. Mech. Comput. Sci. 98(2), 67–76 (2018). https://doi.org/10.26577/jmmcs-2018-2-400
81. D. Draghicescu, A. Caranica, A. Vulpe, O. Fratu, in *2018 International Conference on Communications (COMM)*. Crypto-mining application fingerprinting method, (2018), pp. 543–546. https://doi.org/10.1109/ICComm.2018.8484745
82. F. Gomes, M. Correia, in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. Cryptojacking detection with cpu usage metrics, (2020), pp. 1–10. https://doi.org/10.1109/NCA51143.2020.9306696
83. G. Gomes, L. Dias, M. Correia, in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. Cryingjackpot: network flows and performance counters against cryptojacking, (2020), pp. 1–10. https://doi.org/10.1109/NCA51143.2020.9306698
84. F. Naseem, A. Aris, L. Babun, E. Tekiner, S. Uluagac, in *Network and Distributed Systems Security (NDSS) Symposium 2021, February 21-25, 2021 Virtual*. Minos*: a lightweight real-time cryptojacking detection system, (2021). https://dx.doi.org/10.14722/ndss.2021.24444
85. S. S. Ali, A. ElAshmawy, A. F. Shosha, in *Proceedings of the International Conference on Security and Management (SAM)*. Memory forensics methodology for investigating cryptocurrency protocols, (2018), pp. 153–159
86. A. Gangwal, M. Conti, Cryptomining cannot change its spots: detecting covert cryptomining using magnetic side-channel. IEEE Transactions on Information Forensics and Security. 15, 1630–1639 (2020)
87. J. D'Herdt, Detecting crypto currency mining in corporate environments. Technical report, SANS (2018). https://www.sans.org/reading-room/whitepapers/threats/paper/35722
88. E. L. Jamtel, in *2018 11th International Conference on IT Security Incident Management IT Forensics (IMF)*. Swimming in the monero pools, (2018), pp. 110–114. https://doi.org/10.1109/IMF.2018.00016
89. A. Swedan, A. N. Khuffash, O. Othman, A. Awad, ed. by A. Abuarqoub, B. Adebisi, M. Hammoudeh, S. Murad, and M. Arioua. Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (ACM, Amman, 2018), pp. 23–12310. https://doi.org/10.1145/3231053.3231076
90. H. N. C. Neto, M. A. Lopez, N. C. Fernandes, D. M. F. Mattos. Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking, vol. 75, (2020), pp. 121–131. https://doi.org/10.1007/s12243-019-00744-4
91. M. Caprolu, S. Raponi, G. Oligeri, R. D. Pietro, Cryptomining makes noise: detecting cryptojacking via machine learning. Comput. Commun. 171, 126–139 (2021). https://doi.org/10.1016/j.comcom.2021.02.016
92. A. Pastor, A. Mozo, S. Vakaruk, D. Canavese, D. R. López, L. Regano, S. Gómez-Canaval, A. Lioy, Detection of encrypted cryptomining malware connections with machine and deep learning. IEEE Access. 8, 158036–158055 (2020). https://doi.org/10.1109/ACCESS.2020.3019658
93. Y. Feng, D. Sisodia, J. Li, in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security ASIA CCS '20*. Poster: content-agnostic identification of cryptojacking in network traffic (Association for Computing Machinery, New York, 2020), pp. 907–909. https://doi.org/10.1145/3320269.3405440
94. i Muñoz J.Z., J. Suárez-Varela, P. Barlet-Ros, in *5th IEEE International Symposium on Measurements & Networking, M&N*. Detecting cryptocurrency miners with NetFlow/IPFIX network measurements (IEEE, Catania, 2019), pp. 1–6. https://doi.org/10.1109/IWMN.2019.8804995
95. V. Veselý, M. Zádník, How to detect cryptocurrency miners? by traffic forensics! Dig. Investig. 31, 100884 (2019). https://doi.org/10.1016/j.diin.2019.08.002
96. F. Iglesias, T. Zseby, Analysis of network traffic features for anomaly detection. Mach. Learn. 101(1-3), 59–84 (2015). https://doi.org/10.1007/s10994-014-5473-9
97. SupportXMR, List of all Monero Pools (2021). http://moneropools.com/ Accessed 21 Apr 2021
98. Slushpool, What is Vardiff (variable difficulty algorithm)? (2021). https://help.slushpool.com/en/support/solutions/articles/77000433929-what-is-vardiff-variable-difficulty-algorithm- Accessed 21 Apr 2021
99. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011)
100. node-cryptonote-pool Developers, Mining pool for CryptoNote based coins such as Bytecoin and Monero. GitHub (2020). https://github.com/zone117x/node-cryptonote-pool
101. nodejs-pool Developers, nodejs-pool. GitHub (2020). https://github.com/Snipa22/nodejs-pool
102. Stichting Cuckoo Foundation, Cuckoo Sandbox - automated malware analysis (2021). https://cuckoosandbox.org/ Accessed 19 Apr 2021
103. K. Khade, X. Lin, WebCobra malware uses victims' computers to mine cryptocurrency (2018). https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/webcobra-malware-uses-victims-computers-to-mine-cryptocurrency/ Accessed 21 Apr 2021
104. N. Hyvärinen, NRSMiner updates to newer version (2019). https://labsblog.f-secure.com/2019/01/03/nrsminer-updates-to-newer-version/ Accessed 21 Apr 2021
105. Nanopool, Claymore-XMR-Miner (2017). https://github.com/nanopool/Claymore-XMR-Miner/blob/master/README.md Accessed 21 Apr 2021
106. OpenSSL Software Foundation, OpenSSL: Cryptography and SSL/TLS Toolkit (2020). https://www.openssl.org/docs/man1.0.2/man1/ciphers.html Accessed 21 Apr 2021
107. mineXMR.com, High performance Monero mining pool (2021). http://minexmr.com/ Accessed 21 Apr 2021
108. J. Grunzweig, Large scale Monero cryptocurrency mining operation using XMRig (2018). https://unit42.paloaltonetworks.com/unit42-large-scale-monero-cryptocurrency-mining-operation-using-xmrig/ Accessed 21 Apr 2021
109. CoinIMP, CoinIMP FREE JavaScript Mining (2020). https://www.coinimp.com/ Accessed 21 Apr 2021
110. PublicWWW, PublicWWW Source Code Search Engine (2020). https://publicwww.com/ Accessed 21 Apr 2021
111. Check Point, April 2019's most wanted malware: cyber criminals up to old 'trickbots' again (2019). https://www.checkpoint.com/press/2019/april-2019s-most-wanted-malware-cyber-criminals-up-to-old-trickbots-again/ Accessed 21 Apr 2021
112. Check Point, October 2019's most wanted malware: the decline of cryptominers continues, as Emotet Botnet expands rapidly (2019). https://www.checkpoint.com/press/2019/may-2019-most-wanted-malware-patch-now-to-avoid-the-bluekeep-blues/ Accessed 21 Apr 2021
113. Check Point, February 2020's most wanted malware: increase in exploits spreading the Mirai Botnet to IoT devices (2020). https://blog.checkpoint.com/2020/03/11/february-2020s-most-wanted-malware-increase-in-exploits-spreading-the-mirai-botnet-to-iot-devices/ Accessed 21 Apr 2021
114. Marathon Studios Inc., AbuseIPDB - IP address abuse reports (2021). https://www.abuseipdb.com/ Accessed 2021 Oct 15
115. Programmer All, Mac sample analysis (2020). https://programmerall.com/article/76021153067/ Accessed 15 Oct 2021

116. Peter "fracpete" Reutemann, Python wrapper for the Weka Machine Learning Workbench (2021). https://pypi.org/project/python-weka-wrapper/ Accessed 19 Apr 2021

Publisher's Note