Computer Communications 171 (2021) 126–139

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Cryptomining makes noise: Detecting cryptojacking via Machine Learning

Maurantonio Caprolu *, Simone Raponi, Gabriele Oligeri, Roberto Di Pietro

Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar

ARTICLE INFO

Keywords:
Machine Learning
Network traffic analysis
Security
Cryptojacking
Cryptocurrencies
Blockchain

ABSTRACT

Cryptojacking occurs when an adversary illicitly runs crypto-mining software over the devices of unaware users. This novel cybersecurity attack, that is emerging in both the literature and in the wild, has proved to be very effective given the simplicity of running a crypto-client into a target device. Several countermeasures have recently been proposed, with different features and performance, but all characterized by a host-based architecture. The cited solutions, designed to protect the individual user, are not suitable for efficiently protecting a corporate network, especially against insiders. In this paper, we propose a network-based approach to detect and identify crypto-clients activities by solely relying on the network traffic, even when encrypted and mixed with non-malicious traces. First, we provide a detailed analysis of the real network traces generated by three major cryptocurrencies, Bitcoin, Monero, and Bytecoin, considering both the normal traffic and the one shaped by a VPN. Then, we propose Crypto-Aegis, a Machine Learning (ML) based framework built over the results of our investigation, aimed at detecting cryptocurrencies related activities, e.g., pool mining, solo mining, and active full nodes. Our solution achieves a striking 0.96 of F1-score and 0.99 of AUC for the ROC, while enjoying a few other properties, such as device and infrastructure independence. Given the extent and novelty of the addressed threat we believe that our approach, supported by its excellent results, pave the way for further research in this area.

1. Introduction

Blockchain actually tries to solve the old problem of *distributed consensus* by exploiting solutions matured from decades of research [1]. The solution coming from Bitcoin's blockchain is particularly interesting: entities participating to the "voting" process should prove to have solved a moderately hard puzzle, the so called Proof-of-Work (PoW) [2]. Indeed, for the vast majority of cryptocurrencies, in order to verify a transaction and to have it added to the distributed ledger, participants are requested to compute a PoW. Computationally solving PoW is referred as *mining*. Over time, the complexity of puzzle solving (typically based on hashing as per Bitcoin and several others [3]) has increased, leading to a rush for deploying more and more powerful systems that nowadays are able to compute more than $40 \cdot 10^{18}$ hashes per second (worldwide hash rate for Bitcoin at the time of writing this paper[1]). ASIC architectures are today guaranteeing the best trade-off between power consumption, terrific hash rate, size, cost, and life-time. The recent adoption of ASIC architectures brings in again the major issue of centralization [4]. Indeed, the huge gap between CPU/GPU and ASIC mining makes the latter the only viable way to participate to the network as a miner. Eventually, this causes centralization since only ASIC-based crypto-miners can participate to the consensus process.

In order to mitigate the above trend, other digital currencies have been created that are actually exploiting different PoW strategies, being therefore ASIC-resistant. For instance, *Monero* is an example of cryptocurrency specifically designed to be mined also with CPU-based architectures. Indeed, *Monero* adopts the *CryptoNight* PoW algorithm where the marginal benefit derived from specialized architectures such as GPU, FPGA, or ASIC does not introduce any significant gain for justifying the adoption of such a hardware. Therefore, mining could also be profitable when performed via CPU-based architectures. The *CryptoNight* algorithm works by filling a segment of cache with random data corresponding to memory addresses, then subsequently hashing the resulting block after reading and writing to those addresses [5].

PoW is becoming a significant source of revenues for the entities participating to the consensus process. This phenomena will grow even more with the increasing number of users joining the digital currency markets. However, while PoW computational requirements are fueling methodologies and techniques to achieve more and more computational power with less energy consumption, new malicious practices involves PoW-offloading to unaware users. Indeed, a very recent cybersecurity attack involves the illicit use of resources from an unaware users to carry out PoW, i.e., *cryptojacking* [6]. This attack mainly

* Corresponding author.
E-mail addresses: mcaprolu@hbku.edu.qa (M. Caprolu), sraponi@hbku.edu.qa (S. Raponi), goligeri@hbku.edu.qa (G. Oligeri), rdipietro@hbku.edu.qa (R. Di Pietro).

[1] https://www.blockchain.com/en/charts/hash-rate.

https://doi.org/10.1016/j.comcom.2021.02.016
Received 31 May 2020; Received in revised form 17 November 2020; Accepted 16 February 2021
Available online 26 February 2021

consists on the unauthorized mining of cryptocurrencies allowing malicious parties to steal resources in terms of CPU, GPU, and memory from a target machine with the aim of effortlessly collecting crypto-wealth. This behavior is gaining momentum for two main reasons: the ease of deployment of crypto-clients; and, the difficulty to detect those crypto-clients. While being a general threat, cryptojacking is becoming particularly critical in Corporate ICT where the vast majority of laptops, desktops, and smartphones are distributed among the employees under a limited (if any) supervision [7]. Indeed, several unauthorized mining activities have already been discovered. Russian nuclear scientists have been arrested for "Bitcoin mining plot" [8], the US government banned a Professor for secretly mining with National Science Foundation supercomputers [9], a former Federal Reserve employee was sentenced to 12 months probation and a $5000 fine after pleading guilty to installing unauthorized software that connected to an online Bitcoin network in order to earn units of the digital currency [10], a Harvard student used 14,000-Core supercomputer to mine Dogecoin [11], the factory lines of Hoya, the leading manufacturer of optical products in Japan, were shut down for three days as hackers tried to establish an unauthorized cryptocurrency mining [12], just to cite a few.

Contribution. The major contributions of this paper are listed below:

- *A new class of attacks*: we define a novel category of attack that subsumes the cryptojacking attack, i.e., the *sponge-attack*, where an adversary (either internal or external) secures a personal profit illicitly exploiting third party computing resources. This category includes known attacks such as cryptojacking, botnets, amplification attacks, etc., and allows a better classification of emerging cybersecurity threats.
- *Network traffic analysis*: we provide a detailed analysis of real network traffic generated by 3 major cryptocurrencies: Bitcoin, Monero, and Bytecoin.
- *Encrypted traffics*: we investigate how VPN tunneling shapes the network traffic generated by Crypto-clients by considering two major VPN brands: NordVPN and ExpressVPN;
- *The Crypto-Aegis Framework*: We propose Crypto-Aegis, a Machine Learning (ML) based framework built over the previous steps to detect several forms of crypto-mining, i.e. solo mining and pool mining and, more in general, the unauthorized presence of a cryptocurrency full-node that could lead to other types of sponge-attack;
- *Comparison*: We compare our results against competing solutions in the literature.

Crypto-Aegis enjoys the following features: (i) *Infrastructure independence*. The analysis is performed at the exit points (edge) of the Corporate network, independently of network size, network layout, and even when multiple layers of encryption are set in place by the attacker, e.g., a VPN is in use; (ii) *Device Independence*. We do not require any modification to the already existing devices adopted by the Corporate employees; (iii) *Multi-adversarial profiles support*. Our solution detects the presence of illicit behaviors via network traffic analysis and independently of the adversarial profiles, i.e., be it an insider or an outsider; (iv) *No clean state required*. Our solution detects the presence of a miner independently of the time the miner started its activities; and, (v) *Effectiveness*. Our solution achieves an F1-score of 0.96 and the AUC of the ROC is greater than 0.99.

Roadmap. The paper is organized as follows. Section 2 resumes the most important contributions related to both cryptojacking analysis and detection. Section 3 provides some background concepts related to the ML tools used in our solution, and presents the most important ML techniques for network traffic classification. Section 4 introduces the scenario and the adversary model. The details related to the measurement setup are depicted in Section 5, while a throughout analysis of the collected network traffic traces is presented in Section 6. Section 7 presents our framework, Crypto-Aegis, introducing our methodology and discussing our results. Finally, a detailed discussion of our results and a comparison with other solutions from the literature is presented in Section 8. Section 9 draws some concluding remarks.

2. Related work

The computational power required to validate and to add blocks to the Bitcoin blockchain has greatly limited the odds that individuals without specialized hardware can provide any contribution to this process. Dedicating small devices (e.g., smartphones, laptops, desktop), or more powerful ones (e.g., workstations, servers), to the mining process would not be worth the cost of the electricity. This discourages users and leaves only a few in the world the opportunity to make contributions and earn the rewards arising. With the advent of other CPU-based cryptocurrencies this scenario has undergone many changes. History repeats itself again. In other ages, by seeing a mine populated by mechanical diggers, the gold digger with the only pick-axe on his shoulders would be forced to find new promising shores. This return to the "gold rush" led to the rediscovery of numerous attacks that had lost meaning with Bitcoin. This type of attacks are identified by the term *cryptojacking*. Hackers, as well as dishonest employees who would like to round off their earnings, "borrow" resources belonging to others to run the mining process. Hand in hand with threats, some solutions have been already proposed, with the aim of implementing countermeasures to mitigate their effects.

2.1. Cryptojacking analysis

In [13], the state-of-the art of crypto-mining attacks have been investigated. By analyzing the malware code, as well as its behavior upon execution, authors examine two common attacks: web browser-based crypto-mining, and installable binary crypto-mining, respectively. Browser-based crypto-mining attacks exploit the JavaScript technology of web-pages, leveraging two web technology's advancements: `asm.js` and `WebAssembly` [14]. Installable binary crypto-mining instead, is possible by using modified versions of the *XMrig* software [15]. The paper analyzes the techniques adopted by cybercriminals to establish a persistence mechanism and avoid detection, and it introduces both static and dynamic analysis, useful to uncover the techniques employed by the malware to exploit potential victims. In [16], the authors present an in-depth study over cryptojacking. The analysis of 853,936 popular web pages led to the identification of 2770 unique cryptojacking samples, of which 868 belonging to Alexa's top 100k ranking websites. A similar solution has been proposed by [17]. The authors propose an approach aiming to identify mining scripts, conducting a large-scale study on the prevalence of cryptojacking in the Alexa's 1 million websites. According to the analysis, on average 1 out of 500 websites hosts a mining script. Numerous works have followed the same direction. In [6], authors conduct measurements to establish the cryptojacking relevance and profitability, wondering whether it should be classified as an attack or as a business opportunity. In [18], 138 million domains have been explored, of which 137 million among com/net/org domains and 1 million coming from the Alexa's Top 1 million list. The analysis shows that the prevalence of browser mining is currently the 0.08% of the analyzed set, a worrying number that should not be underestimated. The Alexa's Top 1 million websites have been taken into account even by [14], in which the authors studied the websites affected by drive-by mining to understand the techniques being used to evade the detection. As a result, 20 active crypto-mining campaigns have been identified. In [17] the authors proposed a 3-phase analysis approach to investigate the cryptojacking phenomenon. They conducted a large-scale study on the Alexa first 1 million websites, finding that approximately 1 out of 500 sites hosts a mining script that immediately starts mining activities when visited.

2.2. Cryptojacking detection

One of the first methodologies used to identify cryptojacking was the analysis of static signatures, as typically done for other types of malware [19]. Several solutions, such as [20] and [21], implement static methods to detect mining activities and blacklist malicious web sites. This approach has been proved ineffective against cryptojacking, because of the usage of obfuscation techniques to evade detection [22]. A first step towards the application of Machine Learning techniques to cryptojacking detection has been made by [23]. The authors present an experimental study in which the dynamic opcode analysis successfully allows the browser-based crypto-mining detection. The proposed model can distinguish among crypto-mining sites, weaponized benign sites (e.g., benign sites to which the crypto-mining code has been injected), de-weaponized crypto-mining sites (e.g., crypto-mining sites to which the `start()` call has been removed), and real world benign sites. In [24], the authors presented a method to detect the browser's malicious mining behavior. Heap snapshot and stack features have been asynchronously extracted and automatically classified using Recurrent Neural Networks (RNNs). With 1159 malicious samples analyzed, the experimental results show that the proposed prototype recognizes the original mining samples with 98% of accuracy if not encrypted, 93% otherwise. In [16], after identifying a set of inherent characteristics of cryptojacking scripts, such as the repeated hash-based computations and the regular call stack, a behavior-based detector called *CMTracker* has been introduced. In [25] the authors proposed CapJack, a machine learning-based detection mechanism able to spot in-browser malicious cryptocurrency mining activities. This solution leverages CapsNet, a machine learning algorithm that mimic biological neural organization. CapJack makes use of system features such as CPU, Memory, disk and network utilization, implementing an host-based solution with a detection rate of 87%. Another approach has been used in [22], where the authors propose an application browser extension, named as CMBlock, able to detect and block mining scripts contained on web pages. The proposed solution combines two different methodologies: blacklisting and a mining behavior detection technique. In [26], the authors provide a hybrid methodology that considers both network traffic and CPU usage to detect mining activities within an host.

All the solutions described in this section are host-based countermeasures designed to detect a mining script running on a single host. For this reason, these solutions are not able to effectively protect a corporate network from the cryptojacking threats, as described in Section 4. In fact, host-based solutions should be installed in every host belonging to the corporate network, with high installation and maintenance costs, not to mention privacy issues. Besides, these solutions use computational resources of the host that they must protect, subtracting them from the business tasks they should be dedicated to.

3. Background

This section provides the reader with background knowledge on the most important techniques used in this paper. The first part contains a description of the Machine Learning techniques that have been adopted in the proposed solution. Then, in the second part we introduce related work that have laid the foundations for the classification of network traffic using Machine Learning algorithms.

3.1. Machine learning tools

Random Forest. Random Forest [27] is an ensemble supervised Machine Learning technique, built as a combination of tree predictors. As an ensemble learning technique, the classification is the result of a decision taken collectively, from a large number of classifiers. The idea behind ensembles classification is based upon the premise that a set of classifiers can provide a more accurate and generalized (thus, less prone to overfitting) classification than a single classifier. With Random Forest, each classifier is a tree, and each tree depends on the values of a random vector independently sampled, but with the same distribution for all the trees in the forest. In detail, for the ith tree, a random vector θ_i is generated, independent of the past random vectors $\theta_1, \ldots, \theta_{i-1}$, but with the same distribution. Each tree i grows using the training set and θ_i, resulting in a classifier h(x, θ_i), where x in an input vector. After a sufficiently large number of trees is generated, each tree casts a unit vote for the most popular class at input x. To guarantee a degree of diversity among the base decision trees, a randomization approach is used, which works well both with bagging and random subspace methods [28]. To generate each single tree with the Random Forest algorithm, several steps are involved. Let N be the number of records in the training set, and M be the number of input variables. The training set (also known as bootstrap sample) is built by sampling N records at random with replacement from the original data. At each node, m variables (with $m \ll M$) are randomly selected out of M. The best split of these m attributes is used to split the node. Once the forest is built, a new instance will run across all the trees in the forest. Each tree provides a classification for the new instance and issues a vote. The majority of the votes will allow to objectively declare the result of the new instance's classification.

k-Fold Cross-Validation. k-Fold Cross-Validation [29] is an accuracy estimation method that allows to evaluate how the results of a model will be generalized to an independent dataset. The main objective of cross-validation methods is to estimate the generalization of a model, that is, to understand its accuracy in the classification of data that it had never seen before (i.e., to avoid the overfitting problem). The method consists in partitioning the dataset into subsets, some of which (e.g., training set) will be used to perform the training of the model, while the remaining ones will be used for validation (e.g., validation set) or for testing (e.g., testing set) purposes. There are two types of cross-validation methods: exhaustive cross-validation methods and non-exhaustive cross-validation methods, respectively. The only difference is the number of subsets generated for the split that are performed. In fact, while the exhaustive cross-validation methods use all possible splitting combinations for training and testing, non-exhaustive methods use only a subset of them. We can therefore say that the non-exhaustive methods are an approximation of the exhaustive ones. K-fold cross-validation is an instance of non-exhaustive methods. In k-fold cross-validation the dataset D is randomly split into k mutually exclusive subsets: $D_1, D_2, \ldots, D_k$ of approximately equal size. The model is trained and tested k times, in particular each time $t \in \{1, 2, \ldots, l\}$ the model is trained on $D - D_t$ and tested on D_t. The cross-validation estimate of accuracy is the number of correct classifications divided by the number of instances in the dataset [29].

3.2. Machine learning techniques for network traffic classification

Network traffic classification has gained more and more attention in the very recent years, having the potential to solve several problems in network management [30,31] (e.g., building network profiles for proactive real time network traffic monitoring and management), as well as in network security [32] (e.g., Machine Learning application for intrusion detection systems or anomaly detection).

The classic approach combines the analysis of the packets header with the payload inspection. Despite the high accuracy of this methodology, the high volume of data to be processed, together with the users privacy issues implied by this approach, pushed the research community to explore different techniques. Moreover, payload inspection is not possible in case of encrypted traffic, that nowadays is practically the norm. A promising research direction explores Machine Learning techniques for both real time IP traffic classification and static offline analysis of previously captured traffic. One of the first work that allowed to understand the effectiveness of Machine Learning algorithms for the classification of network traffic is [33]. The authors make use of unsupervised Machine Learning techniques to automatically classify

traffic flows based on statistical flow characteristics. After studying and evaluating the influence of each feature, including Forward-Pkt-Len-Var, Backward-Pkt-Len-Var, Backward-Bytes, Forward-Pkt-Len-Mean, Forward-Bytes, Backward-Pkt-Len-Mean, Duration, and Forward-IAT-Mean, several traffic traces collected at different locations of the Internet have been used to evaluate the efficiency of the adopted approach. In [34], the authors survey significant Machine Learning-based IP traffic classification solutions proposed in the literature. They highlight that the use of different Machine Learning algorithms for offline traffic analysis (e.g., AutoClass, Expectation Maximization, Decision Tree, Naive Bayes) provides high accuracy (up to 99%) for different Internet applications traffic. In [35], authors evaluate different Machine Learning algorithms for flow-based network traffic classification, in terms of correctness and computational cost. In particular, they investigate the use of three supervised algorithms (i.e., Bayesian Networks, Decision Trees and Multilayer Perceptrons) considering six different classes: Peer-to-Peer (P2P), web (HTTP), content delivery (Akamai), bulk (FTP), service (DNS) and mail (SMTP). Their results show that Decision Trees have both a higher accuracy and a higher classification rate than Bayesian Networks. However, Decision Trees require a larger build time and are more susceptible in the case of incorrect or small amounts of training data. Moreover, they highlight that the amount of training data for a certain traffic class can affect the classification accuracy of both itself and other traffic classes. For this reason, they propose a systematic approach to construct specific training sets that feature the best accuracy results. Regarding the analysis of encrypted network traffic, an early solution is from [36]. The authors use different Machine Learning algorithms (i.e., AdaBoost, Support Vector Machine, Naive Bayesian, RIPPER, and C4.5) to distinguish SSH traffic from non-SSH traffic in a given traffic trace. Their results show that the model generated by C4.5 algorithm outperforms the other ones using flow-based features only. Another contribution in this field is given by [37] and [38]. By using supervised Machine Learning techniques for Android encrypted network traffic analysis, the authors demonstrate that an external attacker can identify the specific actions that a user is performing on his mobile apps. Using a Random Forest classifier, they are able to infer not only the app used by the target user, but also the specific action he performed (e.g., sending an e-mail, posting a message, refreshing the home, and so on) for the most used Android applications, such as Gmail, Facebook, and Twitter, despite the use of SSL/TLS for traffic encryption. Although many interesting techniques have been introduced to carry out network traffic classification tasks, both Decision Trees and Random Forests remain the first choices, due to their outstanding predictive performance (comparable with the best supervised algorithms) and reliable estimation of the importance of features. Not surprisingly, in fact, they are still making the list of the most used algorithms to face such challenge [39–42].

4. Scenario, assumptions, and adversary model

In the following, we describe our reference scenario, the assumptions we make as for the network infrastructure and network traffic classification and, finally, the adversary model.

4.1. Scenario

Fig. 1 shows the details of our reference scenario. We consider a corporate network constituted by several interconnected devices, including one that is controlled by a malicious entity, willing to mine cryptocurrencies without being detected. Our solution should be deployed at the network edge and it involves only an Ethernet connection from the main Corporate Network switch to a server running our Machine Learning algorithm. We observe that our solution requires interventions neither on the employee devices, nor on the already existing network infrastructure. Moreover, our solution can be easily deployed even when there are multiple exit connections between the Corporate Network and the Internet: this can be easily achieved by deploying multiple Ethernet links to collect the data from the Corporate Network exit points. We observe that the above configuration is very conservative with respect to standard commercial solutions. Indeed, in the vast majority of cases, corporate solutions involve hardware for deep packet inspection deployed before the exit point, or even at multiple locations of the network. On the one hand the association between traffic and device is much easier, while on the other hand it requires a significant cost in terms of hardware equipment and deployment. In our solution, we consider the traffic already aggregated, i.e., affected by IP masquerading/NAT, or even tunneled and re-encrypted by a Virtual Private Network (VPN).

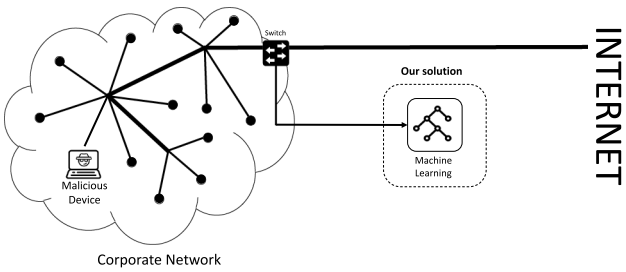

Fig. 1. Network scenario: A Corporate device mines cryptocurrencies controlled by a malicious entity. Crypto-Aegis is constituted by a Machine Learning algorithm classifying the traffic coming from an Ethernet switch at the edge of the Corporate Network.

4.2. Adversary model

We consider two adversary models with respect to the corporate network: (i) *insider*; and, (ii) *outsider*. We assume the insider has direct access to the hardware resources of the company, and therefore, has the opportunity to install new software into it. A typical example is the employee willing to accumulate crypto-wealth by exploiting corporate resources such as CPU, GPU, and network bandwidth. Moreover, we envisage an external adversary (outsider) being able to inject one or more corporate devices with a malicious software for performing unauthorized crypto-mining. Typical examples might be both the increasing number of malware delivering crypto-mining software to unaware users, and websites running crypto-mining Java scripts without the user's consent. Our adversary model (as depicted in Fig. 1) takes into account a corporate device illicitly running crypto-mining-related activities. As previously stated, we stress that our model takes into account a malicious device that might be controlled by either a dishonest employee or by a remote hacker who took over control of the device itself.

There are several strategies to mine without the company consent, an activity that today is really difficult to detect and prevent. This malicious behavior might be implemented by either a *full node* or a *miner*.

- *Full node*. It is a full-featured client of the cryptocurrency infrastructure. It locally stores the whole blockchain and participates to the consensus algorithm, being able to validate all the transactions. The mining activity performed by a full node is called *solo mining* because the process is done independently from other nodes.
- *Miner*. It is a lightweight software that implements a simple worker that receives jobs (i.e., hash computations useful for the PoW) from a third party (i.e., a mining pool). When the mining pool successfully mines a block, both the reward and the fees will be divided among all the participants, proportionally to the computational power offered. This software does not participate in the cryptocurrency protocol and does not require to store a blockchain to work.

The process of installing a crypto-client requires to download the software and, in some cases, additional files. If the client is a full-node, the entire cryptocurrency ledger must also be downloaded. This one-time operation uses the network extensively since a large number of packets is downloaded, being the whole blockchain typically accounting for several GB (for instance, the Bitcoin ledger is currently—Q2 2020—of 260+ GB). In our scenario, we assume that the client (being either a full node or a miner) is already provided with the ledger and, if needed, and does not require any warm-up operations. This assumption, that makes our solution not relying on any application-specific transients, allows us Crypto-Aegis to be effective even when deployed after the malicious device has already started its illicit activity.

Definition. We define *sponge-attack* as the malicious behavior of exploiting third-party hardware and software resources to obtain a personal profit without the authorization of the infrastructure's owner. The sponge-attack illicitly absorbs resources from the targeted infrastructure and makes a pay off out of them in favor of the attacker.

Regarding malicious activities involving cryptocurrencies, the definition of sponge attack is more general than Cryptojacking, which refers only to unauthorized mining activities. The sponge attack, indeed, also refers to any other activity performed unlawfully exploiting someone else's resources. As a result, it also includes configuring a cryptocurrency full node using someone else's resources (network and storage) without permission, even if not used to mine crypto-coins. An example of sponge-attack could be a malicious full node installed on a corporate server to perform a DDoS attack against a cryptocurrency network by using the company's network resources.

The sponge-attack can be implemented by deploying either a full node or a Miner.

Miner. The use of a mining pool software allows to carry out mining activities without installing the heavier full node software. An adversary can use this software to perform a faster and stealthier attack since the targeted device does not need to store the ledger, usually very large. Furthermore, by joining a mining pool, profits are increased even if the available resources are limited.

Full node. Deploying a full node into a network without the administrator consent has significant advantages for the adversary. Firstly, the full node gives to the adversary the capability to perform *solo mining*, if the victim's resources are sufficiently powerful. Moreover, the full node could be used to attack the cryptocurrency's network, by performing double spending attacks, DDoS attacks, Sybil attacks, Eclipse attacks and possibly others.

4.3. Terminology

In the following we refer to different actors and actions by using the fallowing terminology:

- *Crypto-client*: A software illicitly installed in a device belonging to the Corporate Network with the aim of performing the sponge-attack.
- *Standard software*: A software legitimately installed in a device of the Corporate Network.
- *Reference device*: A laptop used to collect network traffic generated by various applications that make up our dataset. This device represents a host legitimately connected to the corporate network.
- *Malicious device*: A reference device that runs crypto-mining software or a cryptocurrency full-node. This device represents a corporate device with mining software installed without authorization.
- *Ingoing flow*: All network traffic generated by an application, coming from the Internet to our reference/malicious device.
- *Outgoing flow*: All the network traffic generated by an application, sent from our reference/malicious device to the Internet.

5. Measurement setup and preliminary considerations

In this section we provide a description of our measurement setup and a preliminary statistical analysis of the collected traces.

Measurement setup. Our measurement setup can be resumed by Fig. 2. We consider two scenarios: Scenario 1 where a VPN tunnel adds an encryption layer to the communication, and Scenario 2 where the client is directly connected to the Internet. In Scenario 1, the malicious device is connected to the Internet through an encrypted VPN tunnel. For our measurements, we used two different well-known VPN brands, i.e., Nord VPN (v. 1.2.0) and Express VPN (v. 1.5.0). At the time of writing this paper, Express VPN features more than 2000 servers in 148 countries while Nord VPN features 5064 servers in 62 countries. We arbitrarily set the VPN exit node to France for all our measurements. Conversely, in Scenario 2, the malicious device is directly connected to the Internet without resorting to any additional encryption layer. The malicious device is a Dell XPS15 laptop running Ubuntu 18.04 (64 bit). As a reference device, we used 2 different laptops; a Dell XPS15 running Windows 10 (64 bit), and a Dell XPS15 running Ubuntu 18.04 (64 bit). All the extracted features are publicly available at [43].

We collected network traffic from three different cryptocurrencies (Bitcoin, Bytecoin and Monero. See Table 1 for the software versions) and three different applications (Skype, YouTube, and standard office applications mixed together) as it follows:

- *Skype*. We run an audio Skype-call and collected all the network traffic from/to the reference device.
- *YouTube*. We collected the network traffic generated by a random YouTube video from/to the reference device.
- *Office network traffic*. We logged the network traffic generated by the reference device while using it for standard office tasks, e.g., e-mail (Outlook for Windows, Gmail Web), web-browsing (Google Chrome, Mozilla Firefox), download and upload of files (Google Drive, Dropbox, Microsoft OneDrive), Microsoft Office365 (Word, Excel), update services (Windows Update, Ubuntu update service) etc.

The above applications have been selected as a reference excerpt of three traffic patterns coming from three different application scenarios that are audio calls, video streaming, and standard office network traffic. We observe how such network traffic categories cover more than 87% of the 2018 global consumer internet traffic [44]. Since our idea is to infer on the presence of a Crypto-client from the network traffic, we considered the network traffic from the above standard applications as the background "noise" hiding the traffic of the Crypto-client. Our goal is to discriminate the flows involving the Crypto-client from the other flows in the network.

It is known that Machine Learning for network traffic classification is biased by several parameters, i.e., features, type of traffic, trace length, network state, etc. One major concern is related to the consistency of the extracted features given the limited trace length. In particular, we paid particular attention to capture packets from clients at steady-state and after the initial sync period was accomplished. Indeed, Crypto-clients require a warm-up period to download the blockchain and validate it. This guarantees that our log excerpts represent a consistent snapshot of a steady-state client either syncing or mining for the blockchain network.

6. Network traffic analysis and patterns

In this section, we start the analysis of the collected network traffic by considering the two network flows: ingoing and outgoing, as explained in the previous section. In order to guarantee a fair comparison between the various scenarios, we extracted the same number of consecutive samples for each network trace, i.e., 4576 samples. This choice is a consequence of the fact that cryptographic mining software used to mine in pools generates a very limited amount of

Table 1
Cryptocurrencies and clients.

Cryptocurrency	Type	Client	Version
Bitcoin	Full node	Bitcoin Core	0.17.0
Bitcoin	Miner	bfgminer	5.5.0
Monero	Full node	Lithium Luna	0.12.3.0
Bytecoin	Full node	Bytecoin Wallet	3.3.2
Bytecoin/Monero	Miner	XMrig	2.8.1

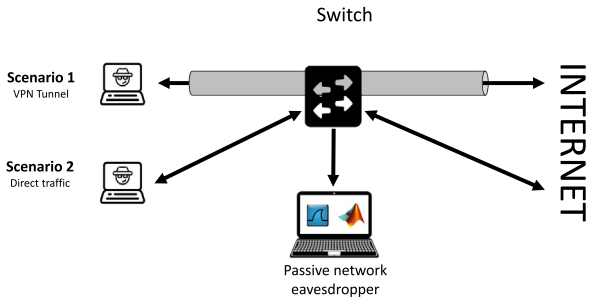

Fig. 2. Measurement setup: We consider 2 different scenarios. The malicious device is connected to the network through a VPN Tunnel (Scenario 1) and the malicious device is directly connected to the Internet (Scenario 2). We adopted one laptop for the mining activities (malicious device), one other laptop for collecting all the in-transit packets and finally, a switch featuring a monitoring port.

traffic—because of the limited amount of interaction needed between the worker and the pool. Hence, in accordance with the best practices for ML, we balanced our dataset (in terms of number of packets) to avoid biasing our data in favor of one of the classes. Indeed, [45] shows that the class imbalances phenomenon hinders the performance of some standard classifiers, including Decision Trees. The same reasoning can be applied to the Random Forest technique, since it was originally designed as a combination of Decision Trees. The improvements that a balanced training set can bring with respect to an unbalanced training set are shown in [46], where the Decision Tree technique applied to the balanced datasets outperforms the same technique applied to the unbalanced ones, both in terms of accuracy (up to +40%) and in terms of ROC curve.

Table 2 shows the network traces we have collected considering the different application scenarios, i.e., Office, Skype, YouTube, Bytecoin, Monero, and Bitcoin. For each scenario, we report the trace duration equivalent to the extracted samples, the quantile 0.5 computed on the interarrival times, and finally the quantile 0.5 computed on the packet sizes. In order to ease the discussion, we refer to each trace by using a sequence of keywords as follows: *[Application][Flow direction][VPN Type]*, where Application can be Office, Skype, YouTube, Bytecoin, Monero, or Bitcoin, Flow direction might be either Ingoing or Outgoing, while VPN Type might be empty (no VPN), Express, or Nord.

Firstly, we observe how considering the same amount of samples involves very different collection time depending on the application scenario, i.e., about 38.37 s for YouTube Ingoing with Express VPN, while about 4598 s for Bytecoin Ingoing. In the following we provide some insight from Table 2:

- *Bytecoin.* Interarrival times are significantly affected by the use of VPN, i.e., time reduction spans from 5 to 10 times. Packet sizes are affected as well, i.e., the increase spans from 2 times to 3 times. It is worth noting that the reduction of the interarrival time with the increasing of the packet sizes involves a reduction of the trace length to guarantee the delivery of the same amount of data.
- *Monero.* VPN tunneling affects interarrival times of Monero depending on the flow. While ingoing flows experience a reduction of the interarrival time, outgoing flows slightly increase their values. Packet size is affected by the same phenomena. While packet size of ingoing flow ramps up from 66 Bytes (No VPN) to 1433 Bytes (Nord), outgoing flows work in the opposite way decreasing from 1242 Bytes (No VPN) to 131 Bytes (Nord).
- *Bitcoin.* Interarrival times are more homogeneous for Bitcoin. Indeed, values span between 180 μs and 300 μs. Nevertheless, we observe that VPN tunneling affects packet size, indeed for both Nord VPN and Express VPN, packet size is becoming significantly larger.

Table 2
Collected traces: Duration, median of interarrival times, and median of packet sizes.

Trace	Trace duration [s]	Int. time median [s]	Pkt. size median [bytes]
Office Ingoing	50.998	0.000118	1434
Office Outgoing	59.502	0.000311	60
Office Ingoing Express	84.460	0.001003	874
Office Outgoing Express	147.281	0.013116	478
Office Ingoing Nord	104.700	0.000265	119
Office Outgoing Nord	105.479	0.000162	1433
Skype Ingoing	146.1	0.018730	136
Skype Outgoing	145.5	0.019988	130
Skype Ingoing Express	92.577	0.020065	518
Skype Outgoing Express	91.911	0.019944	535
Skype Ingoing Nord	87.117	0.020119	169
Skype Outgoing Nord	87.338	0.020734	196
YouTube Ingoing	140.35	0.000001	1434
YouTube Outgoing	896.04	0.001074	54
YouTube Ingoing Express	38.378	0.001848	927.5
YouTube Outgoing Express	816	0.022749	483
YouTube Ingoing Nord	168.48	0.004814	1432
YouTube Outgoing Nord	271.93	0.007282	119
Bytecoin Ingoing	4597.2	0.004673	593
Bytecoin Outgoing	3280.4	0.001130	66
Bytecoin Ingoing Express	729.67	0.000443	706
Bytecoin Outgoing Express	979.43	0.000858	134
Bytecoin Ingoing Nord	1579	0.000803	1432
Bytecoin Outgoing Nord	2011.1	0.000752	119
Monero Ingoing	822.55	0.000450	66
Monero Outgoing	790.58	0.000014	1242
Monero Ingoing Express	197.52	0.000044	890
Monero Outgoing Express	215.45	0.000090	820
Monero Ingoing Nord	445.29	0.000137	1433
Monero Outgoing Nord	404.01	0.000117	131
Bitcoin Ingoing	669.27	0.000600	90
Bitcoin Outgoing	659.45	0.000242	66
Bitcoin Ingoing Express	356.1	0.000383	146
Bitcoin Outgoing Express	389.88	0.000180	146
Bitcoin Ingoing Nord	692.44	0.000502	165
Bitcoin Outgoing Nord	822.29	0.000359	119

Discussion. VPN tunneling tends to squeeze the packets all together and to increase the packet size. Bitcoin is special: VPN tunneling is affecting much less the original traffic pattern although there are some significant variations for the packet size. It is worth noting the differences among the cryptocurrencies when the traffic is collected without VPN tunneling. Interarrival times and packet sizes are very different from each other among the currencies as well as between the ingoing/outgoing flows.

Given the above considerations, we consider more in-depth analysis of the flows in order to subsequently identify the features to be used for the Machine Learning process. Figs. 3 and 4 show quantile 0.05, 0.5, 0.95, minimum and maximum values associated to each collected trace during our measurements. Outgoing flows present packet sizes very different from each other in the range between 100 and 1000 bytes. Only few exceptions fall out of that range, while it is worth noting how quantile 0.5, e.g., the median, changes for each network trace. Moreover we observe that, raw traffic from Bytecoin, Monero, and Bitcoin present almost the same values of quantile 0.05 and 0.95. Interestingly, such values get closer (being characterized by less variations) when their traffic is tunneled through a VPN.

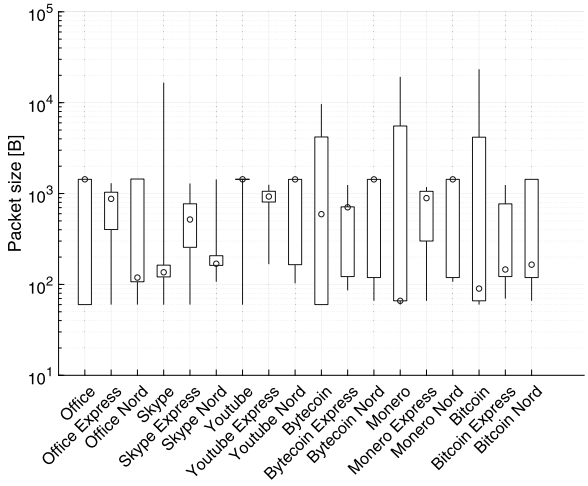

Fig. 3. Packet size for outgoing flows: Candle-sticks represent the minimum, quantile 0.05, quantile 0.95 and the maximum packet size for the outgoing flows while the circles represent quantile 0.5.

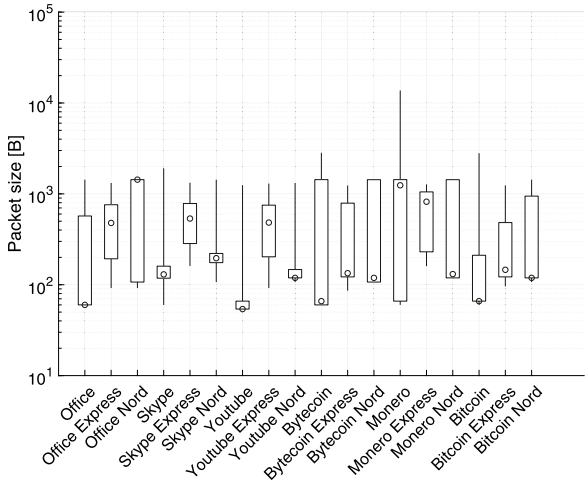

Fig. 4. Packet size for ingoing flows: Candle-sticks represent the minimum, quantile 0.05, quantile 0.95 and the maximum packet size for the ingoing flows while the circles represent quantile 0.5.

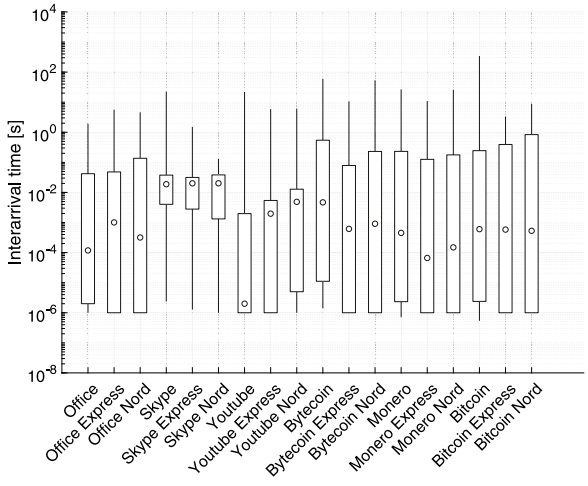

Fig. 5. Interarrival times for ingoing flows: Candle-sticks represent the minimum, quantile 0.05, quantile 0.95, and the maximum interarrival times for the ingoing flows—circles representing quantile 0.5.

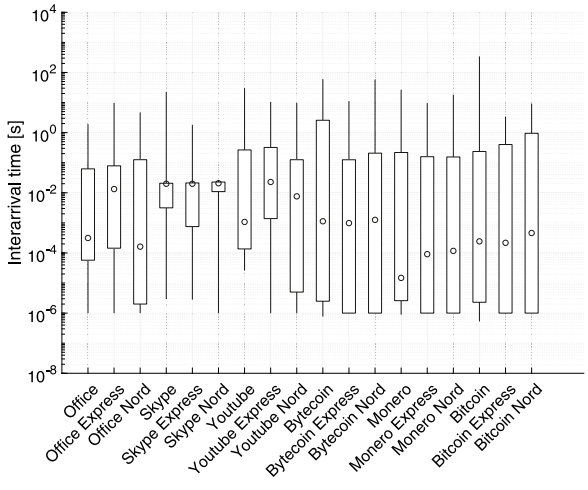

Fig. 6. Interarrival times for outgoing flows: Candle-sticks represent the minimum, quantile 0.05, quantile 0.95 and the maximum interarrival times for the outgoing flows while the circles represent quantile 0.5.

Ingoing flows behave differently from outgoing ones. Packet size spans between closer ranges, i.e., quantile 0.05 and 0.95 are closer with respect to the outgoing flows. Median values are randomly distributed and we do not observe any significant pattern in the VPN tunneling of cryptocurrency clients.

We performed the same analysis for the interarrival times obtained by differentiating the absolute arrival times logged by WireShark. The ingoing flows (Fig. 5) of cryptocurrencies are characterized by very similar values, i.e., almost the same quantile 0.05 and 0.95, although we observe that the median values span between 10^{-2} and 10^{-4} seconds. Similar observations can be drawn by looking at the outgoing flows as depicted by Fig. 6.

7. Traffic classification: The Crypto-Aegis framework

In this section, we discuss our methodology and we present our results. First, we depict a baseline example analysis in Section 7.1, i.e., Bitcoin vs. standard software, introducing all the statistics that will be considered for the subsequent analysis. Then, we introduce the methodology used by our framework by performing two classification experiments. In Section 7.2, we discuss the detection and identification of full nodes, while in Section 7.3 we focus on how to identify miners. The goal of this investigation is to analyze the differences between the network traffic of the considered cryptocurrencies in order to discriminate one from each other. Finally, in Section 7.4 we address the general problem of detecting a cryptographic node in a corporate network, whether it is used exclusively to mine cryptocurrencies or to perform other types of attacks against the cryptocurrency's network classifiable as sponge-attack, e.g., DDoS attack, Eclipse attack, and possibly others.

We implemented all the traffic-classification related tasks in MatLab (R2018a) adopting the Statistics and Machine Learning Toolbox©. Our Crypto-client detection algorithm involves the following steps:

- *Features extraction.* Features identification and extraction are paramount activities to maximize the performance of the classifier. In this work, we consider several features starting from the very standard ones, i.e., interarrival time and packet size. We also consider other derived features with the aim of validating how they affect the final classifier performance, including the moving mean and the moving standard deviation of both the interarrival time and the packet size.
- *k-Fold Cross Validation.* Cross validation is a common practice to average the results of Machine Learning algorithms. It is usually

Table 3
Baseline example: Bitcoin vs. Office scenario.

	Predicted No	Predicted Yes
Actual No	4321	254
Actual Yes	261	4314

performed by defining a random partition of k out of n observations. The partition divides the observations into k disjoint subsamples (or folds), chosen randomly but with roughly equal size. The default value of k is 10.

- *Random Forest (RF).* Before exploring the Random Forest class of algorithms, we considered the performance of different ML algorithms such as Support Vector Machine (SVM), k-nearest neighbors (k-NN), and Naive Bayes. The performance of the aforementioned algorithms were overall worse than those of Random Forest, therefore in the following, we consider RF as the reference algorithm for our detection problem.
  In particular, we adopted the *TreeBagger* MatLab class to implement the RF algorithm. The *TreeBagger* combines the results of many decision trees, which reduces the effects of overfitting and improves generalization. *TreeBagger* grows the decision trees in the ensemble using bootstrap samples of the data.
- *Statistics.* This task involves the generation of statistics from the classifier results. Our statistics include (among others) True Negative (TN), False Positive (FP), False Negative (FN), and True Positive (TP), confusion matrix, etc.

7.1. Traffic classification: a baseline example

In this section, we introduce a simplified version of our methodology considering a binary decision problem. Our goal is to analyze the traffic of a network to determine whether a malicious mining activity is happening. We recall that the traffic collected from the Bitcoin client is related only to the syncing process (being a Full Node), while the one related to the mining process will be considered later on. Moreover, as for the "noise" traffic, we adopted a laptop featuring Windows 10 PRO and performing standard office tasks as discussed in the previous section. We now consider only two network traces from Table 2: Office Outgoing and Bitcoin Outgoing. Each training set is built in a balanced manner, i.e., by including approximately the same number of elements for each class. In this first experiment, we assume the hypothesis H_0: *the current network event has been generated by the Bitcoin client*. We run the 10-Fold cross validation algorithm using the RF algorithm (with a default value of 20 trees) and only two features: interarrival time and packet size. Table 3 shows the confusion matrix associated to the classifier results, i.e., 4314 times Bitcoin is correctly recognized (True Positive — TP) while 4321 times the class Office is correctly recognized (True Negative — TN). The other values refer to False Positive — FP, i.e., 254 observations are wrongly classified as Bitcoin, and False Negative — FN, i.e., 261 observations are classified as not-Bitcoin (Office) while they actually are. Other interesting metrics—that will be used in the remainder of the paper—are the *True Positive Rate* (TPR) = 0.941, i.e., the number of True Positive normalized to the number of actual Bitcoin observations (TP/(TP + FN)), and the *False Positive Rate* (FPR) = 0.059, i.e., the number of False Positive normalized to the number of predicted observation for Bitcoin (FP/(FP + TN)).

FPR and TPR can be used to highlight the classifier performance at different threshold values when the system can accept different levels of false positive values. Fig. 7 shows the Receiver Operating Characteristic (ROC) curve consisting of True Positive Rate (TPR) as a function of False Positive Rate (FPR). Another important metric directly connected to the ROC curve is the so called Area Under the Curve (AUC), i.e., the area under the ROC curve being a value spanning between 0 (worst case) and 1 (best case). As for the ROC curve in Fig. 7, AUC is about 0.971 for both the classes, Bitcoin and Office, respectively.

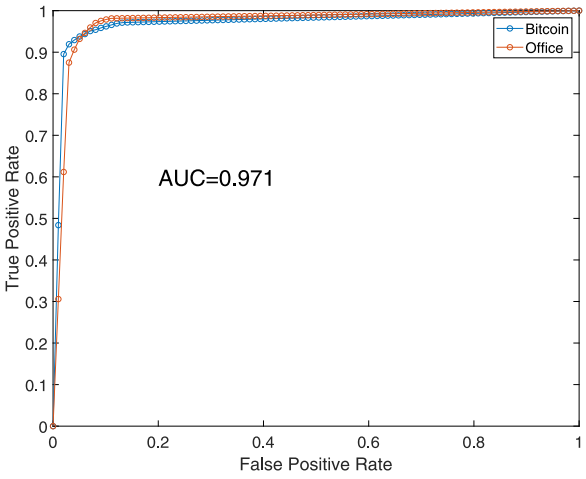

Fig. 7. Receiver operating characteristic (ROC) curve: True Positive Rate as a function of the False Positive Rate. The Area Under the Curve (AUC) is about 0.971 for both the application scenarios.

We now add more features to the current scenario and we analyze the performance of the classifier. Let us define the already (basic) introduced features and the new ones, as follows:

- Interarrival time (δ): the time elapsed between two consecutive packets.
- Packet Size (γ): Packet size associated to each packet.
- Moving mean of δ ($\mu_\delta(w)$): each mean value is calculated over a sliding window of length w across neighboring elements of δ.
- Moving standard deviation of δ ($\sigma_\delta(w)$): each standard deviation is calculated over a sliding window of length w across neighboring elements of δ.
- Moving mean of γ ($\mu_\gamma(w)$): each mean value is calculated over a sliding window of length w across neighboring elements of γ.
- Moving standard deviation of γ ($\sigma_\gamma(w)$): each standard deviation is calculated over a sliding window of length w across neighboring elements of γ.

In order to evaluate the impact of the features on the classification algorithm we used the Mean Square Error (MSE) averaged over all the trees in the ensemble and divided by the standard deviation taken over the trees, for each feature. The larger this value, the more important the feature is in the classification process. Fig. 8 shows the Mean Square Error (MSE) as a function of the moving window size (w) and the different type of features. Firstly we observe that, for this scenario—Bitcoin vs. Office—the most important feature is γ, i.e., the packet size, represented by the red bar. The other features have about the same weights, while it turns out that $w = 5$ is a good trade-off for the window size of the moving mean and the standard deviation.

7.2. Crypto-Aegis: Detection and identification of full nodes

In this section, we consider the traces of Table 2 while parting them into ingoing and outgoing flows. As previously discussed, we consider three main metrics: True Positive Rate (TPR), False Positive Rate (FPR) and the Area Under the Curve (AUC). Our RF classifier has been configured with 20 default trees, the 6 features already introduced in the previous section, and a moving window of 5 observations. Moreover, we consider only Full Node clients; therefore, the observed network traffic will be related to syncing and consensus operations.

Ingoing flows. Fig. 9 shows TPR and FPR for all the cryptocurrencies we have considered in this work. Firstly, we observe how the overall

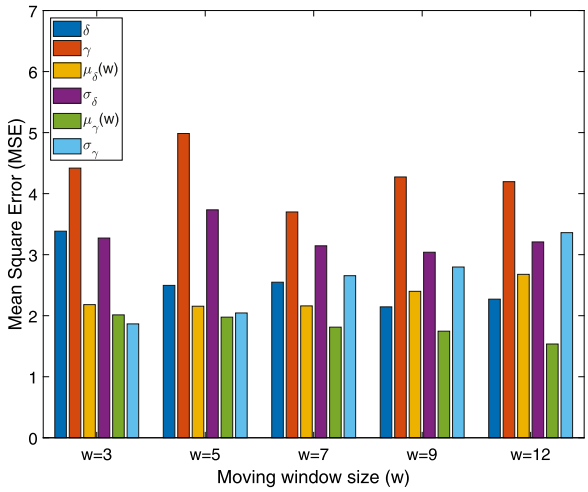

Fig. 8. Mean Square Error (MSE) of the classification results as a function of the features and the moving window size (w).

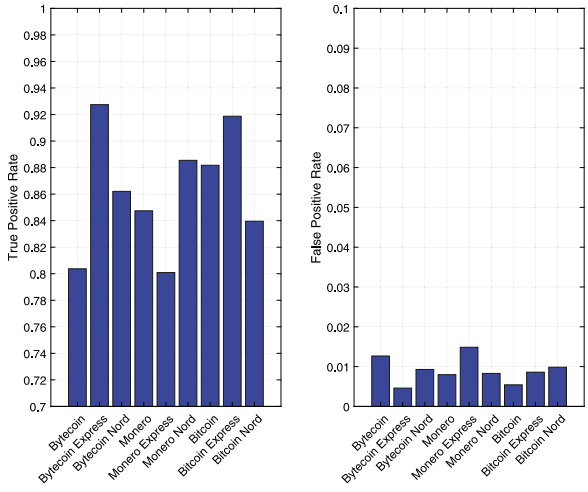

Fig. 9. True Positive Rate and False Positive Rate for ingoing network traffic of a Full Node considering different cryptocurrencies.

results are quite satisfactory, i.e., the mean computed on the TPR and FPR values is about 0.86 and 0.0088, respectively. The best detection performance are achieved over Bytecoin Express (TPR = 0.92, FPR = 0.0047) and Bitcoin Express (TPR = 0.92, FPR = 0.008). Conversely, worst case performance are achieved for:

- Bytecoin (TPR = 0.81, FPR = 0.012). Misclassifications are mainly due to Monero (332 cases - 7%), Bitcoin (106 cases - 2%), and Bytecoin Nord (97 cases - 2%).
- Monero Express (TPR = 0.80, FPR = 0.014). False positive are mainly due to Office Express (381 case - 8%), YouTube Express (148 cases - 3%) and Bytecoin Express (63 cases - 1%).
- Bitcoin Nord (TPR = 0.84, FPR = 0.009). Classification errors mainly come from Monero Nord (251 cases - 5%), Bitcoin Express (217 cases - 4%) and Bytecoin Nord (70 cases - 1%).

Our results prove that ingoing flows (from the Internet to the device) can be used to effectively identify malicious miners inside local networks. In particular, traffic generated by cryptocurrencies clients (without VPN) can be detected with high TPR values, i.e., 0.84, 0.87 and 0.90 for Bytecoin, Monero, and Bitcoin. The adoption of a VPN tunnel does not improve the privacy of the Crypto-client: TPR is increasing for Bytecoin when tunneled through a VPN, while Monero

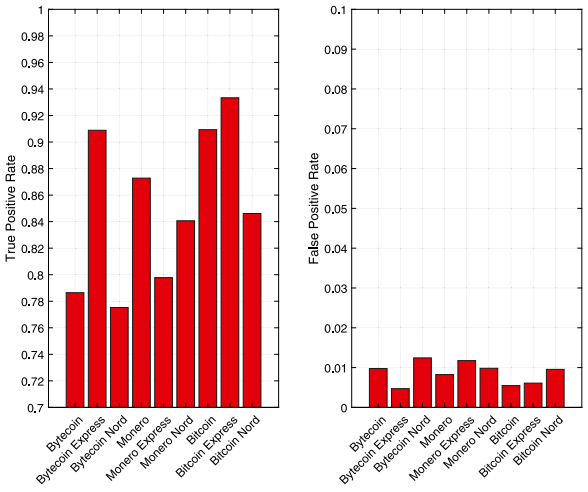

Fig. 10. True Positive Rate and False Positive Rate for outgoing network traffic of a Full Node considering different cryptocurrencies.

and Bitcoin have diverging performance as a function of the adopted VPN brands. Moreover, we observe that worst case performance are due to Crypto-clients misclassified for other Crypto-clients; indeed, there is only one exception: 148 cases of YouTube Express classified as Monero Express. We consider the previous phenomenon as a by-product of the VPN tunneling; indeed, it is reasonable to assume that the same VPN is (slightly) re-shaping different traffic patterns in the same way.

Outgoing flows. Fig. 10 shows TPR and FPR for all the cryptocurrencies we have considered in this work. As for the ingoing flows, we observe how the overall results are, again, quite satisfactory, i.e., the mean value computed on the TPR and the FPR values is 0.85 and 0.008, respectively. The best detection performance are achieved over Bitcoin Express (TPR = 0.93, FPR = 0.006) and Bitcoin (TPR = 0.90, FPR = 0.005). Conversely, worst case performance are achieved for:

- Monero Express (TPR = 0.79, FPR = 0.011). Misclassifications are mainly due to Office Express (563 cases - 12%) and YouTube Express (295 cases - 6%).
- Bytecoin Nord (TPR = 0.77, FPR = 0.012). False positive are mainly due to Monero Nord (281 cases - 6%) and Bitcoin Nord (202 cases - 4%).
- Bytecoin (TPR = 0.78, FPR = 0.009). Classification errors mainly come from Monero (345 cases - 7%) and Bitcoin (147 cases - 3%).

The above considerations prove that outgoing flows (from the device to the Internet) can be used to effectively identify malicious miners inside local networks. Crypto-clients (without VPN) can be detected with TPR of about 0.80 (0.78), 0.84 (0.87) and 0.88 (0.90), for Bytecoin Ingoing (Outgoing), Monero Ingoing (Outgoing) and Bitcoin Ingoing (Outgoing), respectively. As for the ingoing flow analysis, VPN does not significantly increase the privacy of the device: TPR values fluctuate depending on the adopted VPN brand, but they still remain high—and even increase for some cases. One more interesting aspect is that misclassifications happen among cryptocurrencies adopting the same VPN, i.e., a Crypto-client is misclassified as belonging to another cryptocurrency but still using the same VPN; although this is a marginal effect (most frequent case is less than 12%) we can reasonably suspect that VPN tunneling is shaping the traffic patterns; that is why (at least partially) the Machine Learning algorithm experience such type of misprediction.

Area Under the Curve (AUC). As previously introduced, the Area Under the Curve (AUC) is the area under the ROC curve. Its value spans between 0 and 1, and the largest it is, the better the classifier's performance are. Fig. 11 shows the AUC values for the different traces/scenarios in Table 2. For both the ingoing and outgoing flows,

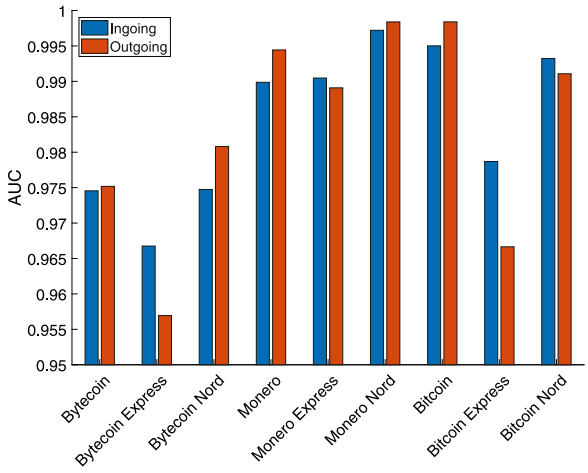

Fig. 11. Area Under the Curve (AUC) for both ingoing and outgoing flows of a Full Node considering different cryptocurrencies.

Table 4
Collected traces from Miners: Duration, median of interarrival times, and median of packet sizes.

Trace	Trace duration [s]	Int. time median [s]	Pkt. size median [bytes]
Bytecoin Ingoing	3935.21	2.41	131
Bytecoin Outgoing	2558.00	0.39	66
Bytecoin Ingoing Express	2486.86	1.47	187
Bytecoin Outgoing Express	1961.86	0.27	122
Bytecoin Ingoing Nord	2513.18	0.37	184
Bytecoin Outgoing Nord	2142.08	0.29	119
Monero Ingoing	15 377.52	13.97	131
Monero Outgoing	10 483.16	6.11	66
Monero Ingoing Express	3658.87	2.46	173
Monero Outgoing Express	3688.58	2.53	122
Monero Ingoing Nord	6184.70	0.34	429
Monero Outgoing Nord	5218.86	0.17	119
Bitcoin Ingoing	8075.92	0.29	125
Bitcoin Outgoing	8014.65	0.37	66
Bitcoin Ingoing Express	2315.13	0.12	181
Bitcoin Outgoing Express	2797.90	0.37	122
Bitcoin Ingoing Nord	4756.34	0.13	178
Bitcoin Outgoing Nord	4644.65	0.19	119

the worst performance are experienced by Bytecoin Express, Bitcoin Express, and Bytecoin. Of course, this does not imply that VPN tunneling is protecting the privacy of the clients as previously discussed. Indeed, we do not observe any significant difference with or without the presence of a VPN tunnel: for all the three cryptocurrencies the AUC values in presence of a VPN tunnel are still very high, i.e., larger than 0.955.

7.3. Crypto-Aegis: Detection and identification of miners

In the following, we consider cryptocurrency clients acting as pure Miners as already introduced in Section 4. Therefore, we consider *bfgminer* for both Bitcoin and *XMrig* for Bytecoin and Monero as illicitly installed clients in a device belonging to a Corporate Network. As for the full node's case, we consider two different VPN, i.e., Express VPN and Nord VPN, and two flows, i.e., ingoing and outgoing.

In the vast majority of cases, the mining task is performed by mining pools [47,48]. In practice, Miners collaborate in pools to lower the variance of their revenue by sharing rewards with a group of other Miners. We consider the same measurement setup of Fig. 2, but we used a Miner as the peer for the blockchain network. As for the analysis of the Full Nodes (Section 7.2), we part all the collected traces into two subsets: ingoing and outgoing flows.

Table 4 resumes the network traces we have collected when considering a Miner as a client: for each trace, we report the duration, the median of both the interarrival times, and the packet sizes. Firstly, we reduce all the traces to the same amount of packets to guarantee a fair classification process for the RF algorithm: we choose 832 packets. We observe how the number of packets from the Miners is less than the one of the Full Nodes: this is mainly due to the fact that Miners generate much less traffic, as it turns out by comparing the interarrival times from Table 4 with the ones from Table 2. Moreover, it is worth noting the case of Monero: the interarrival times are decreasing from 13.97 (Ingoing) and 6.11 (outgoing) seconds to less than 3 s (Express), or 1 s (Nord) when VPN tunneling is activated. Conversely, VPN does not significantly affect packet size with one exception: Monero Ingoing Nord experiences a packet size of 428 Bytes, while the packet size of Monero Outgoing with no VPN is 66 Bytes.

As for the previous case, we consider True Positive Rate (TPR) and False Positive Rate (FPR) as our reference metrics.

Ingoing flows. Fig. 12 shows the TPR and the FPR for different Miner clients when their ingoing flow is compared with the ingoing flows of Office, Skype, and YouTube (same traces from the Standard software of Table 2). Firstly we observe how raw traffic (not tunneled through a VPN) is better identified: Bytecoin, Monero, and Bitcoin turn out to have TPR values equal to 0.993, 0.983, and 0.985, respectively, while FPR values are equal to 0.0018, 0.0004, and 0.0011, respectively.

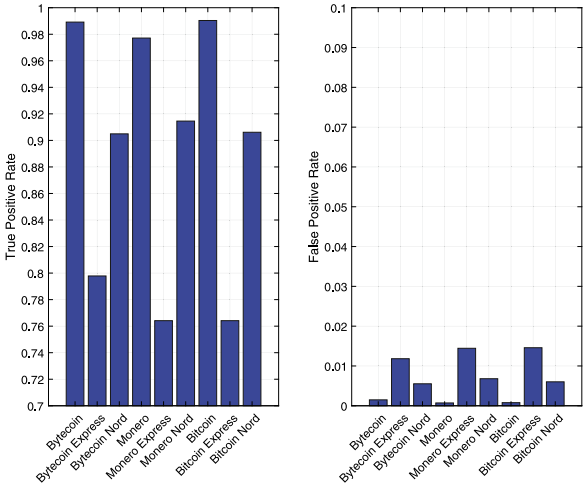

Fig. 12. True Positive Rate and False Positive Rate for ingoing network traffic of Miners of different cryptocurrencies.

Worst case performance are due to Bytecoin Express, Monero Express and Bitcoin Express:

- Bytecoin Express (TPR = 0.79, FPR = 0.011). Misclassifications are mainly due to Monero Express (36 cases - 4%) and Bitcoin Express (10 cases - 1%).
- Monero Express (TPR = 0.77, FPR = 0.015). False positive are mainly due to Bytecoin Express (56 cases - 6%).
- Bitcoin Express (TPR = 0.77, FPR = 0.013). Classification errors mainly come from Monero Express (102 cases - 12%) and Bytecoin Express (68 cases - 8%).

Outgoing flows. Fig. 13 shows TPR and FPR associated to the outgoing flows for all the considered Miner clients compared with Office, Skype, and YouTube network traces. The mean values associated to TPR and FPR are 0.946 and 0.003. The best detection performance are achieved with Bytecoin, Monero, and Bitcoin without VPN tunneling as in the previous case (Ingoing flow analysis). As in this case (ingoing flows), VPN tunneling affects the privacy of the client decreasing the

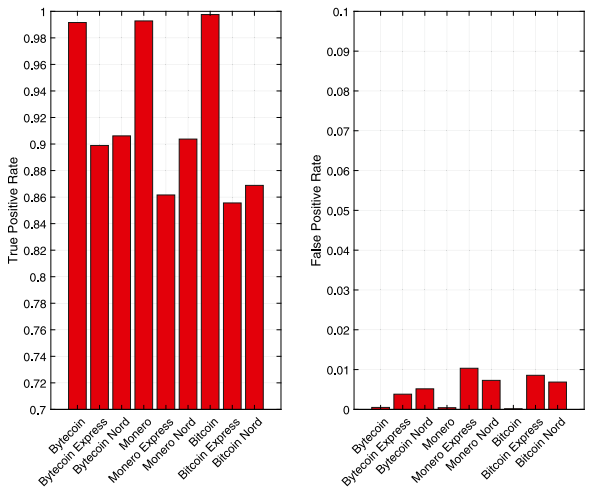

Fig. 13. True Positive Rate and False Positive Rate for outgoing network traffic of Miners of different cryptocurrencies.

identification performance, and in the following, we comment on the worst cases:

- Monero Express (TPR = 0.86, FPR = 0.009). Misclassifications are mainly due to Bitcoin Express (53 cases - 6%) and Bytecoin Express (20 cases - 2%).
- Bitcoin Express (TPR = 0.86, FPR = 0.008). False positive are mainly due to Monero Express (53 cases - 6%) and Bytecoin Express (13 cases - 1%).

We highlight how—for both the Outgoing and Ingoing cases—misclassification are only due to cryptocurrencies, i.e., the vast majority of the False Positive are due to other cryptocurrencies and not to Standard software traffic. Finally, it is interesting to observe how VPNs are more effective to protect the privacy of the Miners respect to the Full Node scenario. This is mainly due to the fact that the overall traffic sent/received by each Miner is significantly less than that one of Full Nodes and it is buried in the VPN-encrypted traffic.

7.4. Crypto-Aegis: Sponge-attack detection

In this section, we consider the more general binary decision problem of detecting the presence of a Crypto-client in the scenario of Fig. 1. Therefore, we assume the hypothesis H_0: *the current network event has been generated by a Crypto-client*. We consider the traces from the Standard software previously introduced, i.e., Office, Skype, and YouTube, and we test them against all the traces involving a Crypto-clients. As a preliminary step, we pre-process all the traces and uniform their lengths to obtain two classes: the first one constituted by Office, Skype, and YouTube traces (evenly distributed); and, the second one constituted by all the traces from the Crypto-clients (Bytecoin, Monero, and Bitcoin) with and without the VPN tunnels. Tables 5 (6) shows the confusion matrix associated to the previously introduced binary classifier assuming the ingoing (outgoing) network traffic. Firstly, we observe that True Positive sums up to 290 728, i.e., the number of times Crypto-clients are correctly identified. Moreover, the classifier correctly identifies 288 452 events as network traffic from Standard software. Nevertheless, there are 10 255 and 7979 false positive and false negative events, respectively. The results on the outgoing traffic are characterized by similar outstanding performance, i.e., TP = 276 187, TN = 275 969, FP = 9370, and FN = 9152.

Fig. 14 shows the Receiver Operating Characteristic (ROC) curve consisting of the True Positive Rate (TPR) as a function of the False Positive Rate (FPR) assuming only the ingoing flows. We also report:

Table 5
Sponge-attack detection: Ingoing traffic classification.

	Predicted No	Predicted Yes
Actual No	288 452	10 255
Actual Yes	7979	290 728

Table 6
Sponge-attack detection: Outgoing traffic classification.

	Predicted No	Predicted Yes
Actual No	275 969	9370
Actual Yes	9152	276 187

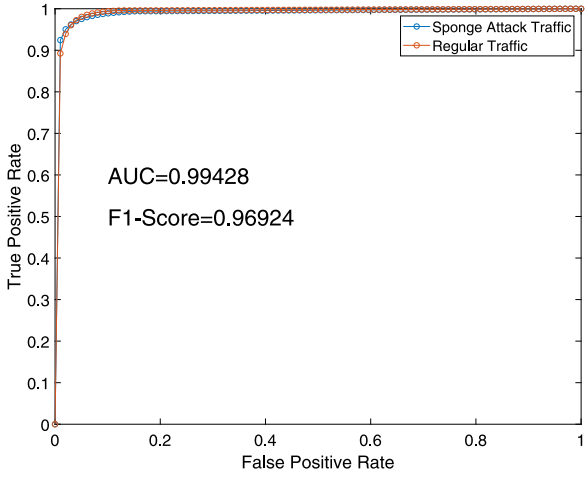

Fig. 14. Sponge-attack detection for Ingoing flows: True Positive Rate as a function of the False Positive Rate assuming the traffic as generated by two classes: Crypto-clients (Miners and Full Nodes) vs. standard applications.

the Area Under the Curve (AUC), being it equal to about 0.99428 for the ingoing traffic; and, the F1-score — given by the harmonic average of the *precision* (TP/(TP + FP)) and the *recall* (TP/(TP + FN)). F1-score is a measure of the accuracy: it is equal to 1 when there is perfect both precision and recall, while it approaches 0 when conditions worsen.

Fig. 15 shows the Receiver Operating Characteristic (ROC) curve consisting of the True Positive Rate (TPR) as a function of the False Positive Rate (FPR), assuming only the outgoing flows. As for the previous case, we report both the AUC and the F1-score.

Time to detect the Crypto-client. Recalling Fig. 8, we observe that the detection time strictly depends on the number of packets requested by the feature generation algorithm; in particular, the moving mean and the moving standard deviation. Full Nodes are characterized by very short interarrival times, i.e., worst case is equal to 0.022749 s (median value for YouTube Outgoing Express), and therefore, waiting for 5 subsequent packets (moving window size used throughout this paper) involves a really short period of time, i.e., less than 120 ms. Even the longest window size depicted in Fig. 8, i.e., 12 packets, requires less than 300 ms. Conversely, Miners require more time for being identified due to their stretched interarrival times. We observe that there are specific Crypto-client configurations from Table 4 that are characterized by large interarrival times, i.e., Bytecoin Ingoing and Monero Ingoing. For such cases, assuming a moving window size w equal to 5 consecutive packets, 12.05 and 69.85 s are required, respectively.

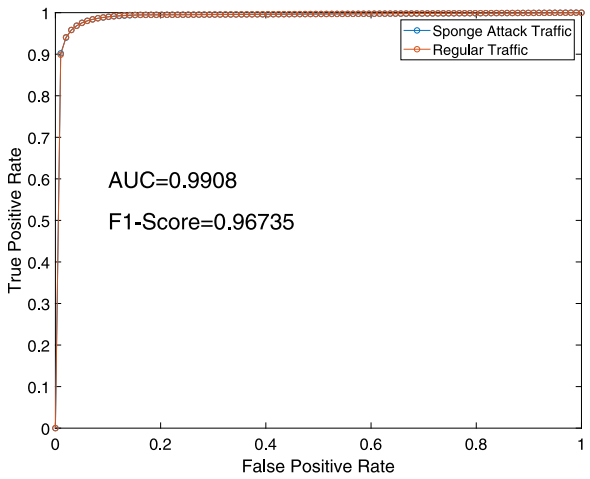

Fig. 15. Sponge-attack detection for Outgoing flows: True Positive Rate as a function of the False Positive Rate assuming the traffic being generated by two classes: Crypto-clients (Miners and Full Nodes) vs. standard applications.

8. Discussion and comparison with other solutions

In this section, we discuss the results shown in this paper, while also comparing our Crypto-Aegis framework against competing solutions existing in the literature.

Full Node. Our solution turned out to be very effective for the detection and identification of Full Nodes in local networks. The RF algorithm is able to independently detect and identify Full Nodes by leveraging either ingoing and outgoing flows. TPR and FPR metrics have similar values although we observe a few exceptions related to the worst cases: Bytecoin Ingoing and Monero Express Ingoing (TPR > 0.8), and Bytecoin Outgoing and Bytecoin Nord Outgoing (TPR > 0.79). Conversely, FPR are always less than 0.015, guaranteeing an extremely low number of potential false alarms.

Miner. Miners' detection performance are similar to the previous case, although we observe that TPR values for the outgoing flows are (in general) greater than the ones for the ingoing flows. This is mainly due to worse anonymization performance achieved by Express VPN for all the ingoing flows. More in general, we highlight the outstanding performance of the classifier when the crypto-clients' network traffic is not tunneled through a VPN. Finally, we stress that, for both the above scenarios (Full Nodes and Miners), false positives are mainly due to cryptocurrencies predicted as other cryptocurrencies. This significantly mitigates the more general problem of the number of false negative when considering the detection of a Crypto-client at large: since FP are still due to other cryptocurrencies, the detection algorithm could be still considered successful (although not being able to fully identify the Crypto-client).

To the best of our knowledge, our contribution is the first one to leverage Machine Learning techniques to detect Crypto-clients by analyzing the network traffic—though ML techniques have been extensively used to effectively detect anomalies in network traffic such as malware, intrusion detection, etc. In this paper we have proved that network traffic generated by Crypto-clients is characterized by pattern anomalies with respect to standard applications such as Skype, YouTube, and standard software used during typical office tasks.

Naïve solutions [20,21,49] dealing with pure web-based mining (cryptojacking) involve blacklisting, a set of malicious URLs reported as suspicious from different sources, i.e., Twitter, blogs, etc. The idea mainly resorts to the installation of a plugin for the web-browser and by monitoring the connections, preventing those belonging to the blacklist. Blacklisting is neither effective nor efficient since it suffers from many false negatives due to URL randomization and requires the installation of third-party software in all the corporate devices. Other solutions involve to monitor the CPU throttle and asking for extra permissions to the web-browser when a process requires high CPU usage [50]. While monitoring the CPU is a promising solution, such a technique potentially suffers from high false positive rates due to the difficulty of discriminating between the miners and other CPU demanding processes, such as videogames. Another interesting technique has been recently proposed in [14]: authors combined multiple techniques to eventually identifying cryptographic operations and inferring on the execution of a miner.

Table 7
Comparison between our solution and the ones in the literature.

	Detection		Identification		Adversary model	
	Full node	Miner	Full node	Miner	Insider	Outsider
Blacklisting [20,21,49]	✗	✓	✗	✗	✗	✓
CPU-throttle monitoring [50]	✗	✓	✗	✗	✗	✓
MineSweeper [14]	✗	✓	✗	✗	✗	✓
CapJack [25]	✗	✓	✗	✗	✗	✓
CMTracker [16]	✗	✓	✗	✗	✗	✓
Crypto-Aegis	✓	✓	✓	✓	✓	✓

We observe that while the above solutions can be adopted for the detection of Miners under certain conditions, they do not take into account Full Node detection. For this reason, the *solo* mining activity, as well as other types of attacks described in Section 4.2, are not recognized by existing solutions. Moreover, none of these solutions can be used to identify the cryptocurrency used by the attacker. Finally, we highlight that all the host-based solutions can be only adopted for the detection of outsider adversaries delivering the crypto-jacking attacks to the users browsing the web. Indeed, none of the above solutions can be used for detecting insider adversaries, i.e., malicious corporate employee willing to run the attack from inside the Corporate Network by having full rights and control of their devices, i.e., laptops, desktops, servers, etc. Table 7 wraps up on the comparison between our solution and the existing ones from the literature.

9. Conclusion

In this paper, in the context of unauthorized crypto-mining activities, we have first proposed a novel attacker model (*sponge-attack*) that subsumes the attacker models present in the literature (cryptojacking). Then, we have introduced Crypto-Aegis, an ML based framework that is able to detect and identify crypto-mining activities related to the sponge-attack class. Crypto-Aegis enjoys several features; in particular: (i) it is infrastructure independent; (ii) it is device independent; (iii) it supports multi-adversarial profiles; (iv) it does not require a clean state to operate; and, (v) it is highly effective (e.g., F1-score of 0.96 and an AUC for the ROC greater than 0.99). Moreover, we have also proved that the Crypto-Aegis framework is resilient to the adoption (by the adversary) of VPN tunneling. The quality and viability of the achieved results, superior to competing solutions in the literature, combined with the novelty of the introduced attacker model, pave the way for further research in this domain.

CRediT authorship contribution statement

Maurantonio Caprolu: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing. **Simone Raponi:** Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing. **Gabriele Oligeri:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing, Funding acquisition. **Roberto Di Pietro:** Conceptualization, Investigation, Methodology, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This publication was partially supported by awards NPRP11S-0109-180242, NPRP12S-0125-190013, and NPRP12C-0814-190012 from the QNRF-Qatar National Research Fund, a member of The Qatar Foundation, and by awards HU.BW.723008.S077.HBKU (Innovation - Cycle 1) from Hamad Bin Khalifa University. The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of the QNRF. The authors would like to thank the anonymous reviewers and the AE that, with their insightful comments, helped increasing the quality of the paper. Open Access funding provided by the Qatar National Library.

References

[1] S. Haber, W.S. Stornetta, How to time-stamp a digital document, J. Cryptol. 3 (2) (1991) 99–111, http://dx.doi.org/10.1007/BF00196791.

[2] F. Tschorsch, B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies, IEEE Commun. Surv. Tutor. 18 (3) (2016) 2084–2123, http://dx.doi.org/10.1109/COMST.2016.2535718.

[3] I.M. Ali, M. Caprolu, R. Di Pietro, Foundations, properties, and security applications of puzzles: A survey, ACM Comput. Surv. 53 (4) (2020) http://dx.doi.org/10.1145/3396374.

[4] B. Cohen, Pow change and key reuse, 2018, https://www.getmonero.org/2018/02/11/PoW-change-and-key-reuse.html. (Accessed: 2021-02-18).

[5] J. Kendzicky, Monero (XMR) analysis, 2018, https://medium.com/@jkendzicky16/monero-xmr-analysis-746cf0f656b1. (Accessed: 2021-02-18).

[6] S. Eskandari, A. Leoutsarakos, T. Mursch, J. Clark, A first look at browser-based cryptojacking, in: 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2018, pp. 58–66, http://dx.doi.org/10.1109/EuroSPW.2018.00014.

[7] R. Di Pietro, S. Raponi, M. Caprolu, S. Cresci, New Dimensions of Information Warfare, Vol. 84, Springer International Publishing, 2021, p. 251, http://dx.doi.org/10.1007/978-3-030-60618-3, part of the Advances in Information Security book series.

[8] BBC.com, Russian nuclear scientists arrested for 'bitcoin mining plot', 2018, https://www.bbc.com/news/world-europe-43003740. (Accessed: 2021-02-18).

[9] Bitcoin Magazine, US Government bans professor for mining bitcoin with a supercomputer, 2014, https://bitcoinmagazine.com/articles/government-bans-professor-mining-bitcoin-supercomputer-1402002877/. (Accessed: 2021-02-18).

[10] T. Mcenery, Trading bitcoin on federal reserve computers apparently totes uncool, 2017, https://dealbreaker.com/2017/01/bitcoin-federal-reserve-scandal/. (Accessed: 2021-02-18).

[11] CCN.com, Harvard student uses 14,000-core supercomputer to mine dogecoin, 2014, https://www.ccn.com/harvard-student-uses-14000-core-supercomputer-mine-dogecoin/. (Accessed: 2021-02-18).

[12] A. Lucian, Japan's Hoya corporation suffers cyber attack computers used for cryptocurrency mining, 2019, https://beincrypto.com/japans-hoya-corporation-suffers-cyber-attack-computers-used-for-cryptocurrency-mining/. (Accessed: 2021-02-18).

[13] A. Zimba, Z. Wang, M. Mulenga, N.H. Odongo, Crypto mining attacks in information systems: An emerging threat to cyber security, J. Comput. Inf. Syst. (2018) 1–12.

[14] R.K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, G. Vigna, Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, in: CCS '18, ACM, New York, NY, USA, 2018, pp. 1714–1730, http://dx.doi.org/10.1145/3243734.3243858, URL: http://doi.acm.org/10.1145/3243734.3243858.

[15] L. Kessem, XMRig: Father zeus of cryptocurrency mining malware?, 2018, https://securityintelligence.com/xmrig-father-zeus-of-cryptocurrency-mining-malware/. (Accessed: 2021-02-18).

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

[17] M. Musch, C. Wressnegger, M. Johns, K. Rieck, Thieves in the browser: Web-based cryptojacking in the wild, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, in: ARES '19, ACM, New York, NY, USA, 2019, pp. 4:1–4:10, http://dx.doi.org/10.1145/3339252.3339261, URL: http://doi.acm.org/10.1145/3339252.3339261.

[18] J. Rüth, T. Zimmermann, K. Wolsing, O. Hohlfeld, Digging into browser-based crypto mining, in: Proceedings of the Internet Measurement Conference 2018, in: IMC '18, ACM, New York, NY, USA, 2018, pp. 70–76, http://dx.doi.org/10.1145/3278532.3278539, URL: http://doi.acm.org/10.1145/3278532.3278539.

[19] M. Alaeiyan, S. Parsa, M. Conti, Analysis and classification of context-based malware behavior, Comput. Commun. 136 (2019) 76–90, http://dx.doi.org/10.1016/j.comcom.2019.01.003, URL: http://www.sciencedirect.com/science/article/pii/S0140366418300410.

[20] Dr.Mine, 2018, https://github.com/1lastBr3ath/drmine. (Accessed: 2021-02-18).

[21] MinerBlock, 2018, https://github.com/xd4rker/MinerBlock. (Accessed: 2021-02-18).

[22] M.A. Razali, S. Mohd Shariff, Cmblock: In-browser detection and prevention cryptojacking tool using blacklist and behavior-based detection method, in: H. Badioze Zaman, A.F. Smeaton, T.K. Shih, S. Velastin, T. Terutoshi, N. Mohamad Ali, M.N. Ahmad (Eds.), Advances in Visual Informatics, Springer International Publishing, Cham, 2019, pp. 404–414.

[23] D. Carlin, P. O'Kane, S. Sezer, J. Burgess, Detecting cryptomining using dynamic analysis, in: 2018 16th Annual Conference on Privacy, Security and Trust (PST), 2018, pp. 1–6, http://dx.doi.org/10.1109/PST.2018.8514167.

[24] J. Liu, Z. Zhao, X. Cui, Z. Wang, Q. Liu, A novel approach for detecting browser-based silent miner, in: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018, pp. 490–497, http://dx.doi.org/10.1109/DSC.2018.00079.

[25] R. Ning, C. Wang, C. Xin, J. Li, L. Zhu, H. Wu, Capjack: Capture in-browser crypto-jacking by deep capsule network through behavioral analysis, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1873–1881, http://dx.doi.org/10.1109/INFOCOM.2019.8737381.

[26] T.P. Khiruparaj, V. Abishek Madhu, P.R.K. Sathia Bhama, Unmasking file-based cryptojacking, in: J.D. Peter, S.L. Fernandes, A.H. Alavi (Eds.), Intelligence in Big Data Technologies—beyond the Hype, Springer Singapore, Singapore, 2021, pp. 137–146.

[27] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, http://dx.doi.org/10.1023/A:1010933404324.

[28] V.Y. Kulkarni, P.K. Sinha, Random forest classifiers: a survey and future research directions, Int. J. Adv. Comput. 36 (1) (2013) 1144–1153.

[29] R. Kohavi, et al., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Ijcai, Montreal, Canada, 1995, pp. 1137–1145.

[30] C.-N. Lu, C.-Y. Huang, Y.-D. Lin, Y.-C. Lai, High performance traffic classification based on message size sequence and distribution, J. Netw. Comput. Appl. 76 (2016) 60–74, http://dx.doi.org/10.1016/j.jnca.2016.09.013, URL: http://www.sciencedirect.com/science/article/pii/S108480451630220X.

[31] D. Sanvito, I. Filippini, A. Capone, S. Paris, J. Leguay, Clustered robust routing for traffic engineering in software-defined networks, Comput. Commun. 144 (2019) 175–187, http://dx.doi.org/10.1016/j.comcom.2019.06.002, URL: http://www.sciencedirect.com/science/article/pii/S0140366418309873.

[32] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, Z. Jia, A mobile malware detection method using behavior features in network traffic, J. Netw. Comput. Appl. 133 (2019) 15–25, http://dx.doi.org/10.1016/j.jnca.2018.12.014, URL: http://www.sciencedirect.com/science/article/pii/S1084804518304028.

[33] S. Zander, T. Nguyen, G. Armitage, Automated traffic classification and application identification using machine learning, in: The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)L, 2005, pp. 250–257, http://dx.doi.org/10.1109/LCN.2005.35.

[34] T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (4) (2008) 56–76, http://dx.doi.org/10.1109/SURV.2008.080406.

[35] M. Soysal, E.G. Schmidt, Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison, Perform. Eval. 67 (6) (2010) 451–467, http://dx.doi.org/10.1016/j.peva.2010.01.001, URL: http://www.sciencedirect.com/science/article/pii/S0166531610000027.

[36] R. Alshammari, A.N. Zincir-Heywood, Machine learning based encrypted traffic classification: Identifying ssh and skype, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–8, http://dx.doi.org/10.1109/CISDA.2009.5356534.

[37] 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) (2016) 114–125, http://dx.doi.org/10.1109/TIFS.2015.2478741.

[38] M. Conti, L.V. Mancini, R. Spolaor, N.V. Verde, Can't you hear me knocking: Identification of user actions on android apps via traffic analysis, in: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, in: CODASPY '15, ACM, New York, NY, USA, 2015, pp. 297–304, http://dx.doi.org/10.1145/2699026.2699119, URL: http://doi.acm.org/10.1145/2699026.2699119.

[39] Y. Fang, Y. Xu, C. Huang, L. Liu, L. Zhang, Against malicious ssl/tls encryption: Identify malicious traffic based on random forest, in: Fourth International Congress on Information and Communication Technology, Springer, 2020, pp. 99–115.

[40] C. Wang, T. Xu, X. Qin, Network traffic classification with improved random forest, in: 2015 11th International Conference on Computational Intelligence and Security (CIS), 2015, pp. 78–81.

[41] B. Yamansavascilar, M.A. Guvensan, A.G. Yavuz, M.E. Karsligil, Application identification via network traffic classification, in: 2017 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2017, pp. 843–848.
[42] A.B. Mohd, S. bin Mohd Nor, Towards a flow-based internet traffic classification for bandwidth optimization, Int. J. Comput. Sci. Secur. (IJCSS) 3 (2) (2009) 146–153.
[43] CRILab, Features extracted from the collected network traces., 2019, https://drive.google.com/open?id=18elZOAYyCO8FabNkuby-wiYF-FD6hPG_.
[44] CISCO, Consumer internet traffic 2016-2021, 2018, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. (Accessed: 2021-02-18).
[45] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intell. Data Anal. 6 (5) (2002) 429–449.
[46] G. Weiss, F. Provost, The Effect of Class Distribution on Classifier Learning: An Empirical Study, Technical Report, 2001.
[47] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, E.W. Felten, Sok: Research perspectives and challenges for bitcoin and cryptocurrencies, in: 2015 IEEE Symposium on Security and Privacy, 2015, pp. 104–121, http://dx.doi.org/10.1109/SP.2015.14.
[48] M. Rosenfeld, Analysis of bitcoin pooled mining reward systems, 2011, CoRR abs/1112.4980. URL: arXiv:1112.4980.
[49] CoinBlockerLists, 2019, https://gitlab.com/ZeroDot1/CoinBlockerLists. (Accessed: 2021-02-18).
[50] T. Mursch, How to find cryptojacking malware, 2018, https://badpackets.net/how-to-find-cryptojacking-malware/. (Accessed: 2021-02-18).