Detecting cryptocurrency miners with NetFlow/IPFIX network measurements

Jordi Zayuelas i Muñoz
UPC-BarcelonaTech
Barcelona, Spain
jordi.zayuelas@est.fib.upc.edu

José Suárez-Varela
UPC-BarcelonaTech
Barcelona, Spain
jsuarezv@ac.upc.edu

Pere Barlet-Ros
UPC-BarcelonaTech
Barcelona, Spain
pbarlet@ac.upc.edu

***Abstract*—In the last few years, cryptocurrency mining has become more and more important on the Internet activity and nowadays is even having a noticeable impact on the global economy. This has motivated the emergence of a new malicious activity called cryptojacking, which consists of compromising other machines connected to the Internet and leverage their resources to mine cryptocurrencies. In this context, it is of particular interest for network administrators to detect possible cryptocurrency miners using network resources without permission. Currently, it is possible to detect them using IP address lists from known mining pools, processing information from DNS traffic, or directly performing Deep Packet Inspection (DPI) over all the traffic. However, all these methods are still ineffective to detect miners using unknown mining servers or result too expensive to be deployed in real-world networks with large traffic volume. In this paper, we present a machine learning-based method able to detect cryptocurrency miners using NetFlow/IPFIX network measurements. Our method does not require to inspect the packets' payload; as a result, it achieves cost-efficient miner detection with similar accuracy than DPI-based techniques.**

Keywords— Cryptojacking detection, cryptocurrency mining, machine learning, NetFlow measurements

I. Introduction

Following the price spike of cryptocurrencies during the end of 2017 and the beginning of 2018, many people saw the opportunity of making money by mining cryptocurrencies. Cybercriminals also noticed this opportunity and started to use compromised machines to mine cryptocurrencies at no cost. This cybersecurity threat, known as cryptojacking, may have very bad impact on the performance of affected machines and has become one of the most important threats in the cybersecurity field, even surpassing ransomware in some cases [1] [2] [3]. Cryptocurrency mining consumes a lot of computing resources, and consequently it can also decrease the life expectancy of the machines compromised. Moreover, it implies a noticeable energy consumption. All this motivates the necessity of detecting possible malicious miners connected to our network.

In cryptocurrency mining, all the miners compete to find the solution of a very hard problem that is necessary to complete and record all the new currency transactions. These transactions are grouped in blocks, and only the first miner that finds the right solution for a block is rewarded with some amount of the currency involved in the transactions. This means that a miner with few resources typically would not be able to solve the problem earlier than the others and, thereby, it would never obtain a reward. For this reason, it is more frequent to create pools of miners that work in collaboration to join their efforts and have higher chance to be the first obtaining the solution. Thus, they can split then the rewards obtained. Based on this, it is possible to detect miners by using lists of IP addresses from known cryptocurrency mining pools or using information in DNS queries involving domain name resolutions of such pools. However, these detection methods are only able to detect miners connecting to known servers, but there are still some ways to bypass them. For instance, connecting to unknown servers, using VPN services or using DNS over TLS to encrypt the data in DNS records.

Other method more reliable is to apply Deep Packet Inspection (DPI) to look for mining activity [4]. The most commonly used protocol by mining pools is Stratum, which is transmitted over TCP in plain text using JSON-RPC messages. This way DPI can be used to inspect the traffic content and identify well-known signatures that are present in these connections. The main problem of this method is that typically it is not feasible to perform DPI over all the traffic in real network deployments, since it is too expensive and networks may not have the necessary monitoring infrastructure to capture and process all the traffic. Moreover, this method can also be bypassed by miners using VPNs to connect to the server, or miners using unknown protocols.

In order to address this problem, we propose a novel machine learning-based method that uses NetFlow/IPFIX flow measurements to perform flow-level detection of mining traffic. Contrary to DPI-based techniques, this method has very low resource consumption given that it only needs to process a reduced amount of data contained in NetFlow reports. Moreover, NetFlow/IPFIX is a well-known protocol that is widely deployed in real-word networks. This makes it more feasible to implement this solution without requiring any upgrade in existing infrastructures.

II. Data Overview

A. Stratum

The data we are interested in is the one coming from cryptocurrency mining protocols. Stratum is the protocol used by most of the mining pools to communicate between the miners and the pool servers. It has a limited set of messages that servers can send to miners, and another set of requests that miners can send to servers.

This protocol is implemented on top of TCP, and there is no well-known port related to it. This makes it useless using the server's port to apply port-based classification to detect Stratum traffic.

Stratum works over JSON-RPC, and there are five messages commonly present in the communication between clients and servers [5]:

 DOI 10.1109/IWMN.2019.8804995

- Mining.subscribe: Used at the start of the connection to indicate to the server that the client is ready to start mining.
- Mining.authorize: Sends identification information to the server.
- Mining.submit: The client sends a found share to the server.
- Mining.notify: Indicates to the client the data that is going to be used to mine.
- Mining.set_difficulty: Short message used by the server to indicate the difficulty level used when mining.

Figure 1 shows the TCP layer data transmitted between a miner (red) and a mining pool server (blue) during a mining session. Looking at the transmitted data we can see that it is transmitted in clear text. If we take a close look at the transmitted data, we can see that most of the data is transmitted in the server's mining.notify method. Also, we can observe that the client sends very few data in comparison to the server.

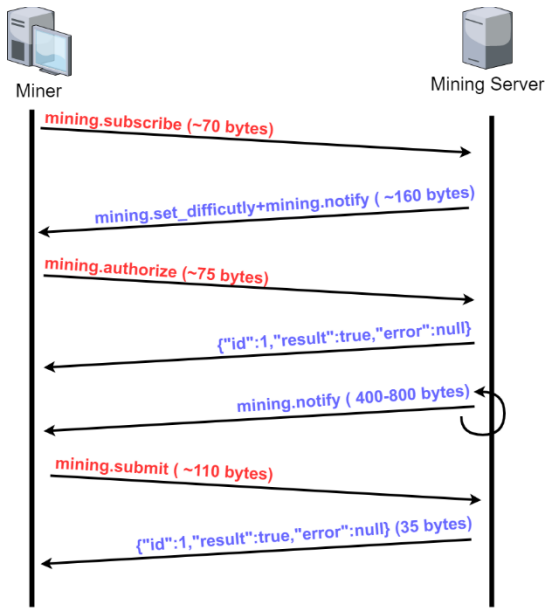


Fig. 1. TCP connection between a miner (red) and a pool server (Blue) using the Stratum protocol

Note that although Stratum is the most used protocol to mine, there are other protocols like *getblocktemplate* or a variant used by the Miner gate application. However, they share similar methods with different names and generate almost identical traffic patterns.

B. NetFlow/IPFIX

NetFlow/IPFIX is a protocol that performs flow-level network measurements by aggregating all the traffic matching the same 5-tuple. This is the source and destination IPs, the TCP/UDP ports, and the protocol over IP.

Note that the traffic reported by NetFlow is unidirectional. The client-server and the server-client packets of a same connection will form two different flow measurement records.

Each flow record contains the following information: Source and destination IPs, source and destination ports, next hop, input and output SNMP interface, number of packets transmitted, number of bytes transmitted, start and end timestamps, TCP flags, transport protocol, type of service, source and destination autonomous system, and source and destination address masks [6].

However, we do not need all this data for our purpose. We only use information about the traffic flows (identifiers, traffic volume and time). The rest of the data related to routing and network administration can be ignored.

Data like the number of packets and bytes are absolute values related to the total duration of the flows. Thus, in order to create better identification patterns, we compute the traffic volume values with respect to the flow times. This is Packets/second, bits/second. Also, we process the average size of packets (bits/packet).

One fact that characterizes Stratum connections (Fig. 1) is that they are quite asymmetric. While the server transmits a lot of data to clients, the client sends a very small amount of data to the server. In order to exploit this asymmetry, we combine the NetFlow records corresponding to outbound and inbound flows. This allows us to get more valuable information about the bidirectional connections. As a result, two new fields were also added to the machine learning model: packets inbound/packets outbound and bytes inbound/bytes outbound. This helps to observe the difference of traffic volume transmitted in both directions.

III. DATASET

A. Stratum Data

The first thing we noticed when starting to collect the flow measurements was that the Stratum traffic generated flows on a steady but slow way. Each flow took between 5 and 30 minutes to finish. In order to collect enough data to train a machine learning model, we decided to generate mining traffic on our own, since we do not have access to network traffic with a significant amount of mining traffic.

This way, we performed mining over different cryptocurrency servers and captured the traffic. Note that different cryptocurrencies may have some variations on their traffic patterns, and this may have an effect on the features used to classify the flows. For instance, the number of transactions may influence the number of packets sent by the server, the block size may affect the total amount of data transmitted, and the block time to reset all the mining jobs might result in different traffic volume sent from the server.

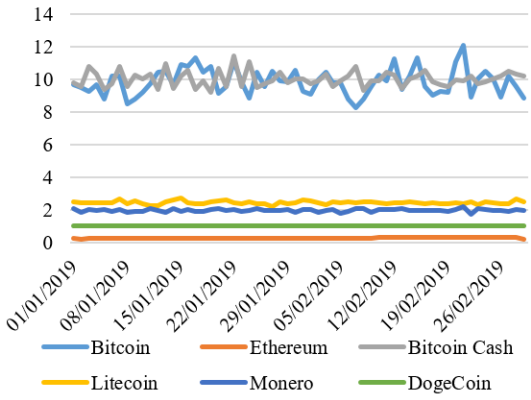
To achieve a representative overview of the mining traffic, we mined five of the most popular cryptocurrencies: Bitcoin, Bitcoin-Cash, DogeCoin, LiteCoin and Monero. Table 1 shows some relevant characteristics of these cryptocurrencies.

Coin	Average Block time	Average Block size	Average per-block transactions	Hashing algorithm
Bitcoin	10 min	787kb	2270	SHA-256
Bitcoin-Cash	10 min	50kb	115	SHA-256
DogeCoin	1 min	11kb	20	Scrypt
LiteCoin	2.5 min	20kb	42	Scrypt
Monero	2 min	16kb	6	Cryptonight

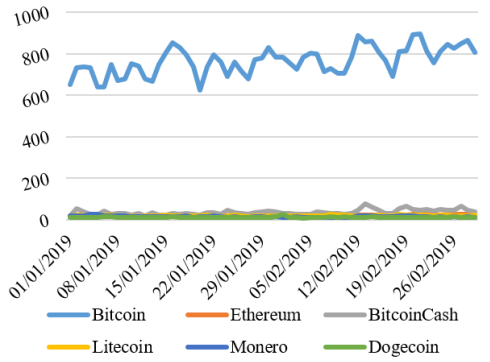
Table 1: Characteristics of the cryptocurrencies in the dataset.

Also, one fact to take into account is that the value of these characteristics is not static, they have slight variations depending on how profitable is the currency at any given time, how many people is making transactions and other arbitrary facts. Although they remain in a relatively small boundary, this still complicates the process of classifying the mining traffic. All the traffic in the dataset has been captured during the course of a month, switching between cryptocurrencies to have more diverse data.

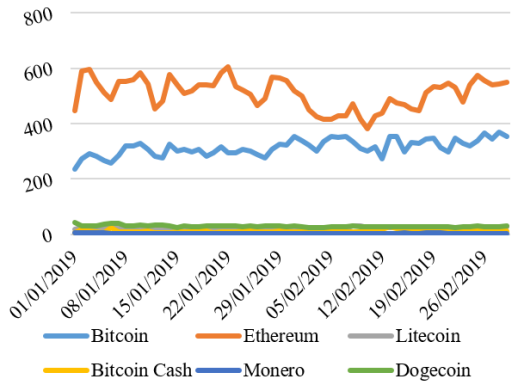
Figure 2 represents these variations in three different plots, which show the average block time, the block size and the number of transactions. Over the course of the two months each of the cryptocurrencies has shown many variations regarding these 3 statistics.



(a) Average Block Time (minutes)



(b) Average Block Size (Kbytes)



(c) Number of transactions

Fig. 2. Evolution of the block time, block size and number of transactions over the course of two months.

In total, we spent between 24 and 72 hours to mine each cryptocurrency and finally we could capture 677 mining flows.

B. *Traffic Capture from a Campus University Network*

To train ML models, we used a traffic capture from a large campus university network. In order to leverage this data to train a supervised machine learning model, we previously identified the flows coming from mining traffic. To this end, we applied DPI over all the traffic and found some flows generated by miners. This was achieved by using two different signatures that allowed us to match methods of the Stratum protocol on the packet's payload. The former signature identifies the server-side methods and the latter one finds client-side methods. Discriminating the Stratum traffic in both directions is relevant for us, since we will build our ML solution around the assumption that flows are asymmetric, so we must know who is the server and who is the client to be able to process the bidirectional flows in the dataset. We show below the two regular expressions we used to implement both signatures:

1) \"method\" ?: ?\"(mining\.notify|mining\.set_difficulty)\""

2) \"method\" ?: ?\"(mining\.authorize|mining\.get_transactions|mining\.subscribe|mining\.submit|getblocktemplate|submitblock)\"

Our dataset contained 1,795,408 TCP flow pairs in total, where 14 were identified as mining traffic associated to three different mining pool servers. We discarded flows with only 1 packet (and duration 0) belonging from failed connections.

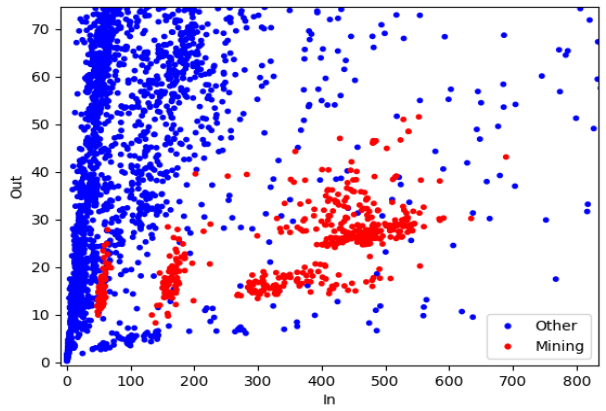
Finally, we trained the machine learning model using a combination of this dataset with the 677 mining traffic samples of Stratum we previously extracted (Sec. III A).

IV. Data Analysis

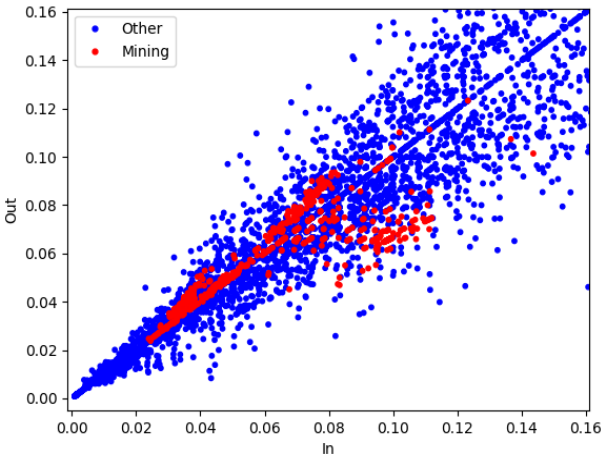
A. *Mining Traffic*

One relevant characteristic of the mining traffic we captured is that all flows are long (in duration) but they have a very small number of packets sent, and almost all the data goes in the direction from the server to the client. Particularly, the server typically sends 20 times more data than the client on average. Indeed, the data sent from the client is very reduced, and the average packet size is very close to the size of a TCP segment without any payload.

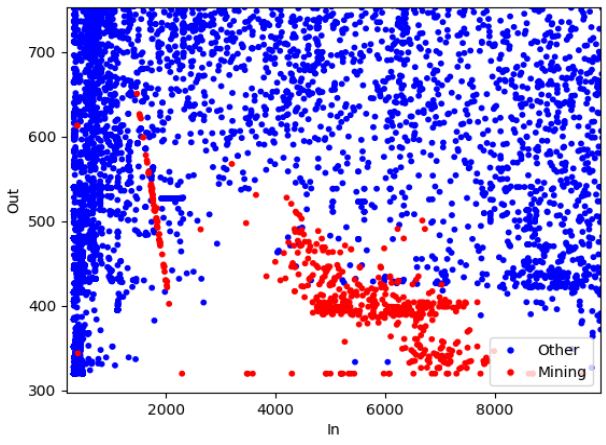
In order to gain more insight into the traffic in our dataset, we process some flow-level traffic statistics that may be quite relevant to differentiate Stratum mining traffic from the rest. These statistics are the per-flow mean bits/second, packets/second and bits/packets. Particularly, we process separately unidirectional flows and combine them in bidirectional flows with inbound and outbound traffic statistics (i.e., server-to-client and client-to-server traffic respectively). In Figure 3, we show scatter plots with these statistics. They show clear differences between flows created by mining applications (in red). Even they are not clear enough to make a trivial classifier, we believe that a machine learning algorithm would be able to create a good classifier able to detect the flows belonging to mining applications with high accuracy.



a) Bits/second



b) Packets/second



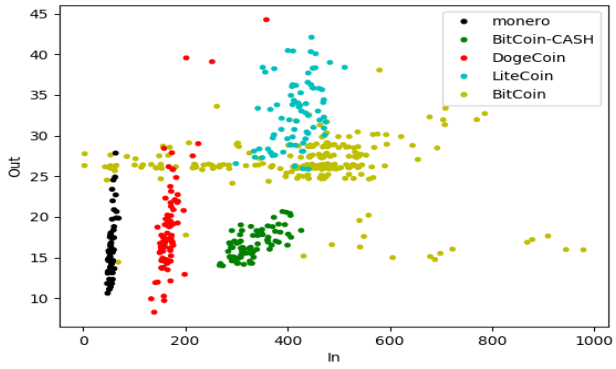
c) Bits/packet

Fig. 3. Comparison between flows created by mining applications (red) and other applications (blue). (a) shows the bits/second, (b) shows the packets/second and (c) shows the bits/packet. The y-axis represents traffic in the client-server direction and the x-axis traffic in the server-client direction.

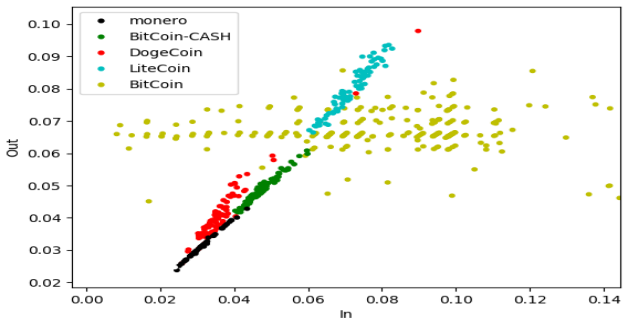
Note that the purpose of these figures is only to illustrate some possible differences between the mining traffic generated by Stratum and other applications. However, these statistics do not necessarily represent the most significant features for the machine learning model to discriminate Stratum traffic flows.

B. Traffic from Different Cryptocurrencies

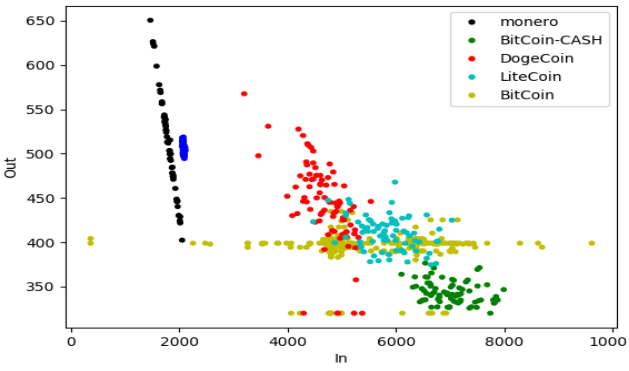
In the data represented in Figure 3, we could also observe some different clusters in the traffic from mining applications. Thus, we further analyzed these points to find some relevant information from these different clusters. Figure 4, represents the previous statistics considering only the points from mining flows and separated by applications mining different cryptocurrencies.



a) Bits/second



b) Packets/second



c) Bits/packet

Fig. 4. Comparison between flows created by applications mining different cryptocurrencies. (a) shows the bits/second, (b) shows the packets/second and (c) shows the bits/packet. The y-axis represents traffic in the client-server direction and the x-axis traffic in the server-client direction.

As we can observe in these figures, there are also clear differences between the traffic patterns of flows related to different cryptocurrencies. This motivated us to create also another model to classify mining flows belonging to different cryptocurrencies.

V. Evaluation

In order to evaluate our ML-based method, we test different Machine Learning (ML) models using input features from NetFlow measurement reports in WEKA [7]. Particularly, these models are trained with the following input features: *(i)* inbound and outbound packets/second, *(ii)* inbound and outbound bits/second, *(iii)* inbound and outbound bits/packet, *(iv)* bits_inbound/bits_outbound ratio and *(v)* packets_inbound/packets_outbound ratio. The ML models tested are the following: Support Vector Machines (SVM), CART, C4.5 decision tree and Naïve Bayes.

Since we have a reduced amount of mining traffic in our dataset, we avoid splitting the dataset into training and evaluation by applying a 10-fold cross-validation to train and test the models. This allows us to better leverage the data for the training and still perform an appropriate evaluation.

We compare the performance of the different ML models with the following well-known statistics for classification problems:

$$Accuracy = (TP+ TN)/(TP+TN+FP+FN)$$

$$Precision = TP/(TP+FP)$$

$$Recall = TP/(TP+FN)$$

Where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

Table 2 shows the results achieved by the different ML models to detect mining traffic. Note that they are binomial classifiers with the following classes: *(i)* Stratum and *(ii)* non-Stratum.

Algorithm	Accuracy	Precision	Recall
SVM	0.99658	0.00000000	0.00000000
CART	0.99996	0.98236331	0.90716612
C4.5	0.99998	0.98009950	0.96254072
Naïve Bayes	0.90863	0.00370975	0.99511400

Table 2: Evaluation results of Stratum traffic detection with different ML models

From these results we can observe that the accuracy results are quite good in general. This is partly due to most of the flows are not related to mining applications. Thus, classifying all the flows as non-Stratum results in high overall accuracy. This is for instance the case of the SVM model, whose recall measurement is zero. On the other hand, all the models based on decision trees (C4.5 and CART) achieved the best results. The best classifier was the C4.5 decision tree, so we decided to keep it as our model.

The resulting C4.5 decision tree not only is very accurate, but also has a very low computational cost. The maximum depth of the tree is 13 leaf nodes. Moreover, in most of the cases it employs only 5 split operations to classify correctly the traffic.

The same methodology is applied to create a ML model able to differentiate the cryptocurrencies mined in our dataset. Particularly, we aim to classify the 5 different cryptocurrencies depicted in Table 1. For this evaluation, we use the accuracy and the average F-Score measurements. This latter metric allows us to combine the results of precision and recall in a single metric:

$$F\text{-}Score = 2 \cdot (Precision \cdot Recall)/(Precision+Recall)$$

Table 3 shows the results of this evaluation. As we can observe, all machine learning models achieve quite good accuracy results. In this case the best classifier is the one based on Naïve Bayes, which achieved an accuracy of 0.963.

Algorithm	Accuracy	Average F Score
SVM	0.912	0.909
CART	0.963	0.963
C4.5	0.967	0.967
Naïve Bayes	0.973	0.973

Table 3: Evaluation results of cryptocurrency classification with different ML models

In Table 4, we show a confusion matrix with the evaluation results achieved by the Naïve Bayes model for each cryptocurrency. As we can observe, the absolute number of flows misclassified is very low.

a	b	c	d	e	f	
77	0	0	0	0	0	a = Bitcoin-Cash
4	297	0	2	0	0	b = Bitcoin
0	1	83	1	0	0	c = DogeCoin
0	2	0	74	1	0	d = LiteCoin
0	0	4	0	54	0	e = Monero
0	0	0	0	0	91	f = Ethereum

Table 4: Confusion matrix with the results of the Naïve Bayes model.

The resulting ML models have demonstrated ability to differentiate between flows generated by Stratum mining traffic and flows created by other applications with high accuracy. This shows the possibility of using ML-based solutions to detect mining applications using only data extracted from Netflow reports. Note that, the precision of our ML models could be further improved by adding mechanisms tracking TCP connections over time. This way, since miners are typically connected all the time, it is possible to maintain records with the flows that were already classified as miners and report alerts only in the cases that the connection times exceed a threshold. This would dramatically reduce the number of false positives generated by the model.

VI. Related Work

Detecting traffic related to cryptocurrency mining in networks is of paramount importance. Specially, with the recent increase of malicious activities such as cryptojacking. As a result, this has attracted a lot of attention from the research community, which has recently devoted some efforts to create mechanisms to detect cryptocurrency mining traffic in networks.

We could find in the literature different approaches to identify traffic related to mining activities. One example is the code analysis of web applications to find some instructions that are often present in mining algorithms [8] [9]. Others propose to analyze the memory consumption in web applications and compare it to the memory used by well-known mining applications [10].

Alternatively, other studies propose to analyze the network traffic and look for traffic patterns commonly present in the traffic related to mining applications [11].

In the context of Machine Learning, we could also find some techniques aiming to detect mining applications at endpoints [12] [13].

In contrast to all previous proposals, our approach is to use ML-models and use only information from flow-level measurements reports of NetFlow/IPFIX in order to achieve accurate detection of mining traffic at limited cost.

VII. Conclusions and Future Work

In this paper, we implemented and evaluated different Machine Learning (ML) models to detect cryptocurrency miners in the traffic using only information from NetFlow/IPFIX flow-level measurement reports. Our evaluation results show that our ML-based method was able to accurately detect the traffic generated by mining applications. The main advantage of our method is that it only needs to process a reduced amount of data from NetFlow reports to classify the traffic. As a result, it shows to be more efficient than other accurate methods based on resource-hungry Deep Packet Inspection techniques.

Also, we showed the possibility to create ML-based models to identify the specific cryptocurrencies that were mined. Particularly, we achieved accurate ML models able to discern among 5 different cryptocurrencies.

As future work, we plan to improve the model using a more extensive dataset containing more flows from mining applications. Also, we plan to extend our cryptocurrency classifier to support a wider range of cryptocurrencies.

Acknowledgment

This work has been supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE).

References

[1] Check Point Blog, "January 2019's Most Wanted Malware: A New Threat Speaks Up," January 2019. [Online]. Available: https://blog.checkpoint.com/2019/02/13/january-2019s-most-wanted-malware-a-new-threat-speakup-linux-crypto-cryptomining/. [Accessed 15 March 2019].

[2] M. Bayern, "Cybersecurity rundown: The 5 most critical threats to businesses in 2018," TechRepublic, 17 July 2018. [Online]. Available: https://www.techrepublic.com/article/cybersecurity-rundown-the-5-most-critical-threats-to-businesses-in-2018/. [Accessed 15 March 2019].

[3] A. Dascalescu, "Here Are the Biggest Cybersecurity Threats to Watch out for in 2018," Heimdal Security, 18 July 2018. [Online]. Available: https://heimdalsecurity.com/blog/biggest-cybersecurity-threats-2018/. [Accessed 15 March 2019].

[4] J. D'Herdt, "Detecting Crypto Currency Mining in Corporate Environments," SANS Institute InfoSec Reading Room white-paper, 2017.

[5] "Stratum Mining Protocol," Slush Pool, 2012. [Online]. Available: https://slushpool.com/help/stratum-protocol/#!/manual/stratum-protocol. [Accessed 17 March 2019].

[6] "NetFlow Export Datagram Format," CISCO, 14 September 2017. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html. [Accessed 18 March 2019].

[7] University of Waikato, " Weka 3 - Data Mining with Open Source Machine Learning Software in Java," University of Waikato, [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/index.html. [Accessed 10 03 2019].

[8] R. K. Konoth et al., "Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense," in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1714–1730.

[9] M. Saad, A. Khormali, and A. Mohaisen, "End-to-end analysis of in-browser cryptojacking," arXiv preprint arXiv:1809.02152, 2018.

[10] J. Liu, Z. Zhao, X. Cui, Z. Wang, and Q. Liu, "A Novel Approach for Detecting Browser-Based Silent Miner," in IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018, pp. 490–497.

[11] A. Swedan, A. N. Khuffash, O. Othman, and A. Awad, "Detection and prevention of malicious cryptocurrency mining on internet-connected devices," in Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, 2018, p. 23.

[12] A. Kharraz et al., "Outguard: Detecting In-Browser Covert Cryptocurrency Mining in the Wild," 2019.

[13] D. Carlin, P. O'kane, S. Sezer, and J. Burgess, "Detecting Cryptomining Using Dynamic Analysis," in 16th Annual Conference on Privacy, Security and Trust (PST), 2018, pp. 1–6.