Cryptojacking Detection in Cloud Infrastructure using Network Traffic

Philip Kwedza
Department of Computer Science Rhodes University
Makhanda, South Africa
kwedzaphilip@gmail.com

Stones Dalitso Chindipha
Department of Computer Science
Rhodes University
Makhanda, South Africa
s.chindipha@.ru.ac.za

2023 International Conference on Electrical, Computer and Energy Technologies (ICECET) | 979-8-3503-2781-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICECET58911.2023.10389593

***Abstract*—Cryptomining is a way to obtain cryptocurrency, by performing computationally complex puzzles in exchange for a reward. To perform this requires expensive specialised hardware to become profitable but most times, this is not viable. Cryptojacking is a cybercrime in which an attacker uses devices to mine cryptocurrency without permission. This attack can be extended to use the resources of networks and cloud infrastructure. This research aimed to develop a model that could detect cryptojacking automatically in a cloud environment, utilising network traffic. The models in this paper solved this by developing a machine learning model that could analyse cryptojacking in a dataset of network traffic from an attacked cloud server.**

***Index Terms*—Cryptojacking, Crypto-mining, Cloud Security, cybercrime, Machine Learning**

I. Introduction

Cryptomining is a way of obtaining cryptocurrency by solving computationally complex puzzles using specialised hardware to make a profit [1]. However, to minimise the cost, a user could distribute the load of the calculations to other devices and therefore does not require specialised hardware. That technique of using unauthorised devices for mining is called cryptojacking [2]. Cryptojacking (also known as drive-by mining) is a cybercrime where an attacker gets a user to perform crypto-mining on their devices without their consent [3].

One way of achieving cryptojacking is by using social engineering to trick the user into downloading a script onto their device and running it in the background [2]. An attacker can also implant a JavaScript or Web-assembly (WASM) script on their website and run the script on the user's web browser. This technique is capable of utilising resources of various networks and cloud infrastructure servers [4].

This type of cryptojacking attack differs from traditional malware as it does not directly endanger users' data. Still, it does exhaust the user's computing resources such as memory and the Central Processing Unit (CPU) [5]. Personal device users are typically unaware of this and might assume that their device was just randomly overheating or being relatively slow at that time.

Most studies, such as those from [6, 3, 7] are centred around detecting cryptojacking on host devices, and therefore there are inadequate methods for detecting cryptojacking within a cloud network environment [4].

Therefore, to address this problem, there needs to be a solution that can detect and stop cryptojacking before cloud server integrity is compromised by using their resources to mine cryptocurrency. An algorithm employing supervised machine learning can use a comprehensive dataset of an attacked cloud server's network traffic. Using that traffic, it should detect cryptojacking independently of other types of cloud-based attacks. An organisation could use a program of this nature to protect its cloud servers and prevent an organisation's resources from being exploited, thus avoiding interference with daily operations.

This paper aims to achieve the following objectives and contribute to the existing body of knowledge:

- Develop a machine learning model that classifies *cryptojacking* using network traffic flows obtained from a cloud environment.
- Generate a dataset that consists of mining traffic on a Virtual Private Server (VPS)

The remainder of this paper is structured as follows: A brief explanation of *cryptojacking* detection as well related studies are explained in **Section II**. In **Section III** there is an overview of the proposed methodology and implementation of the project. **Section IV** discusses the results obtained from the experiments. Lastly, the paper concludes and touches on future plans in **Section V**.

II. Cryptojacking in cloud

This section describes concepts and related studies in *cryptojacking* detection.

A. Network Traffic Classification

Network traffic classification (NTC) is a concept in which network traffic flows can be classified into different patterns. A network flow is all packets that are exchanged between two end-points. By isolating and analyzing each flow, an NTC

979-8-3503-2781-6/23/$31.00 © 2023 IEEE

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.

would be able to classify what particular service a device is using, such as email or streaming, thus making NTC a multi-class classification problem [8]. Cyberattacks have unique patterns, and the NTC could be used for security in a network environment [9].

The most straightforward way to detect and prevent online malicious activity would be to perform deep packet analysis. This is to check packets' content with already known attack patterns for any correlation; software like Snort is a popular tool [10, 6]. Snort is an open-source packet sniffer that is used as a detection engine. Its main advantage lies in that it allows for registering, alerting and responding to known attacks [10]. However, using Snort alone is not viable as it relies on predefined signatures, and attackers would be able to avoid detection by changing the contents of the packets.

Patel *et al.* [10] used Snort as part of their system, but to make up for its vulnerability against unknown attacks, they used the naïve Bayes classifier. The naïve Bayes classifier is a statistical classifier that predicts the probability of a given network event being normal or an intrusion; it can identify malicious activity by matching packets against known behaviours of cyberattacks [10]. Although it can detect many attacks, it can still be bypassed by packet encryption.

Some techniques can classify obfuscated and encrypted data using deep learning. Minehunter [7], which is used explicitly for *cryptojacking* detection in network traffic, is such an example of a deep learning approach. For this project's scope, the NTC will only classify *cryptojacking* network traffic flows, not other services or attacks.

B. Cryptojacking Detection

There have been numerous solutions to detecting cryptojacking in different environments. At first, there were detection methods for detecting cryptojacking in browsers. Konoth *et al.* [11] proposed using *MineSweeper*, a program that could be integrated within a browser. Minesweeper analyses the intrinsic properties of code within web pages, using a Uniform Resource Locator (URL) as input to check for hidden cryptocurrency mining scripts. The technique monitors the most fundamental aspect for crypto-mining to work: the cryptographic computations needed to make valid hashes for transactions.

Wang et al. [12] proposed using Secure in-lined Script Monitors for Interrupting Cryptojacks (SEISMIC). SEISMIC is a browser-based crypto-mining method that uses semantic signature matching to detect cryptojacking for WASM scripts. Wang et al. [12] analysed the cryptocurrency Monero as it is popular amongst attackers. Cryptojacking Monero on browsers is popular as Monero involves the CryptoNight proof-of-work hash algorithm. CryptoNight makes mining on CPUs and GPUs equally as efficient without needing dedicated mining computers. SEISMIC also aims to protect against new cryptojacking attacks, such as web gadgets exploiting counterfeit mining.

Cryptojacking detection is a method in which cryptomining activity can be discovered. This paper describes cryptojacking detection as an NTC problem in which crypto-mining traffic can be classified. There have been a few solutions in which cryptojacking detection, by just using network traffic, has been attempted. Neto *et al.* [13] proposed using *MINEcap*, which is a dynamic online solution that protects an entire network from covert cryptojacking flows. It detects attacks using a machine learning algorithm called super incremental learning on software-defined networks. The primary mechanism used is *Spark Streaming*, a tool used for the online processing of network flows. By monitoring traffic obtained from *Spark Streaming*, should a mining flow be detected, the network traffic controller will block it. The network traffic controller checks both traffic going inside and outside the network. Hence it blocks cryptomining traffic from both sides.

Minos is a lightweight real-time cryptojacking detection system proposed by Naseem *et al.* [14]. *Minos* uses dynamic analysis to detect cryptojacking. Even though it uses dynamic features, the detection of cryptojacking requires that the mining script successfully runs for a certain amount of time [14]. Therefore, to help with that disadvantage, *Minos* was created. *Minos* uses deep learning techniques to detect WASM-based cryptojacking by using an image-based classification to differentiate between benign web pages and cryptomined pages.

An approach that analyses content-agnostic network traffic flows is proposed by Feng *et al.* [3]. The system is designed to detect cryptojacking within a network by evaluating packets from the gateway router. The system first profiles the traffic of crypto-mining activities using a mathematical method called Fast Fourier Transform (FFT) within different periods. It compares that with generated variation vectors and uses neural networks to identify cryptojacking patterns. FFT is an algorithm for faster computation of the discrete fourier transform of a sequence. With FFT, a computer can calculate the various frequencies of time-varying signals [15]. The discrete fourier transform converts a signal's domain (usually in either time or space) to its frequency domain [15].

With the advances in traffic detection algorithms and countermeasures against *cryptojacking*, attackers are developing more advanced techniques to obfuscate network traffic [6]. Zhang *et al.* [7] mentions that common techniques involve using proxies, payload encryption, port replacement and packet padding. Due to such techniques, there is a need to use features that are difficult to confuse. i Muñoz *.et al* [16] proposed using the packets' size and time sequence in a flow as features, as they are difficult to obfuscate.

The next section of this paper will give a very brief overview of the proposed approach.

III. Methodology

This section describes how the proposed *cryptojacking* detection model was designed and developed.

A. Dataset Creation

The dataset was created by using ten Virtual Private Servers (VPS) provided by DigitalOcean. The dataset consisted of

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.

crypto-mining packets developed using *XMRig* to mine on different mining pools. There were 84,344 conversations within the dataset that were categorised as either crypto-mining or normal traffic, with different cryptocurrency coins being mined. To achieve a satisfactory representation of the mining traffic, five crypto-currencies were mined, namely Dogecoin (DOGE), Ethereum Classic (ETC), Ethereum (ETH), Ravencoin (RVN) and Monero (XMR). The network analyser tool *Wireshark* was used to collect the packets. Wireshark was also used to perform Deep Packet Inspection (DPI) on the packets for labelling the dataset.

B. Measuring Model Performance

Multiple metrics can be used to evaluate the performance of machine learning models. The most common metrics used are *accuracy*, *precision*, *recall* and *F1 score*, which combines precision and recall. The metrics used to measure the performance of this model were *F1 score* and Area Under the ROC Curve (AUC). Due to the severely unbalanced nature of the dataset, these metrics are more suited to measuring classifier performance. The F1 score and AUC equations are described in equation 3 and 4, respectively.

$$Precision = \frac{TP}{TP \ + \ FP} \tag{1}$$

$$Recall = \frac{TP}{TP \ + \ FN} \tag{2}$$

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

Using the formulae described in equations 1 and 2, F1 score is described as:

$$F1 \ score = 2 \ \times \ \frac{Precision \ \times \ Recall}{Precision \ + \ Recall} \tag{3}$$

$$AUC = \frac{S_p \ - \ n_p(n_n \ + \ 1)/2}{n_p n_n} \tag{4}$$

S_p indicates the sum of all the positive examples ranked, while n_p indicates the number of positive examples and n_n indicates the number of negative examples.

C. Model Overview

The purpose of the model is to detect *cryptojacking* in cloud network traffic. Machine learning is the process of finding and describing patterns in a supplied dataset. By training the model with the supplied dataset, the Machine learning algorithms will classify whether the network flows of specific IP addresses are using cryptomining or benign traffic. The development of the *cryptojacking* detection model can be described in three stages.

The first stage is obtaining the network traffic data. The data for training the machine learning models were classified as cryptomining and regular traffic, and the cryptomining traffic was further classified into the different coins being mined. The obtained dataset contains network traffic that has been generated from cloud servers.

The second stage is feature engineering and scaling using the network traffic obtained from the previous stage as input. The features used will be discussed further in Section III-D.

The third and final stage is to find and train a model to detect cryptojacking. To measure the performance of the chosen model, the model was compared with other machine learning models using the same data and evaluated using the performance metrics described in Section III-B.

D. Feature Engineering

A certain number of features can predict whether a packet is related to cryptomining. Zhang et al. [7] and i Muñoz *et al.* [16] identified using: *source address*, *destination address*, *source port*, *destination port*, *packet size*, *and the network flow's timestamps* as features that can determine cryptomining within network traffic. Zhang et al. [7] used time-series information of the particular flows to determine which flows are using cryptomining, whereas i Muñoz *et al.* [16] used traditional machine learning algorithms. The cryptomining detection technique used in this paper is based on work done by i Muñoz *et al.* [16]. i Muñoz *et al.* [16] used mean flow-level traffic statistics to differentiate cryptomining traffic from the rest of the traffic.

Figure 1 shows the proposed algorithm to engineer the features.Firstly, identify the unique flows by aggregating all the packets contained in the network traffic matching that have the same 5-tuple.

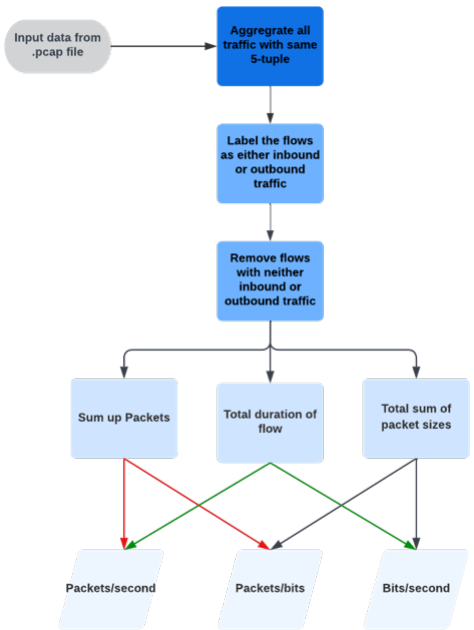
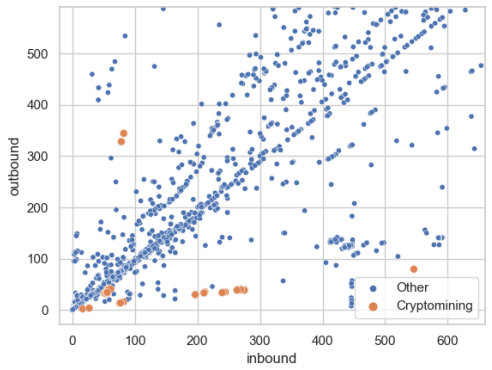
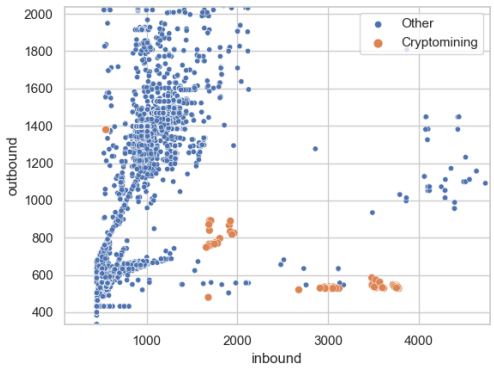


Fig. 1. Proposed feature engineering flow diagram

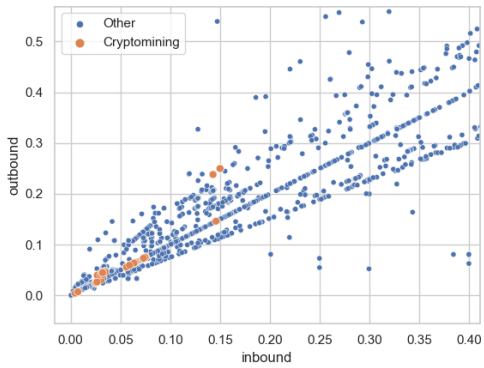
Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.



(a) Bits/Seconds



(b) Bits/Packets



(c) Packets/Seconds

Fig. 2. Flow statistics of cryptomining and normal traffic

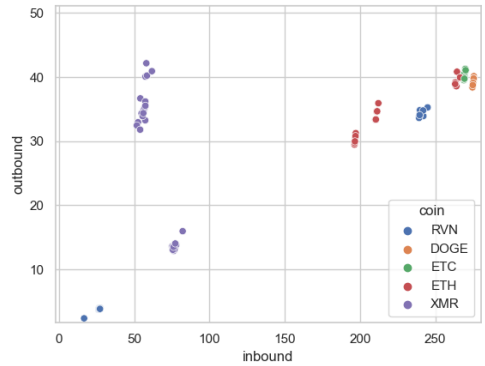
The 5-tuple refers to packets with the same source and destination Internet Protocol (IP) addresses and source and destination port numbers that use the same protocol over IP.

Next, label the flows as either inbound or outbound traffic. Packets leaving the servers are considered outbound traffic, while packets received by the server are considered inbound traffic. Flows that have no inbound or outbound traffic are removed at this stage.

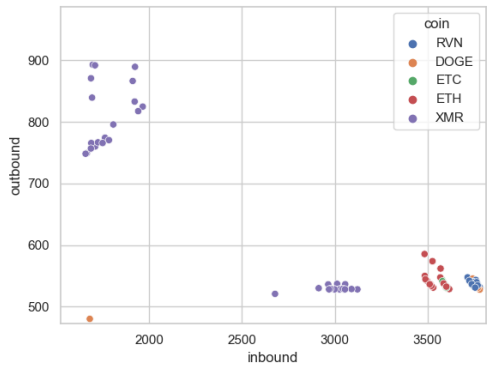
The next phase aggregates the flows to produce the mean flow-level statistics. The number of packets, the total sum of the packet sizes (in bits), and the total time taken by each flow are gathered. The flows containing less than two packets are removed at this stage.

Next, calculate the number of packets per second,bits per second and bits per packet for each flow for both inbound and outbound traffic.

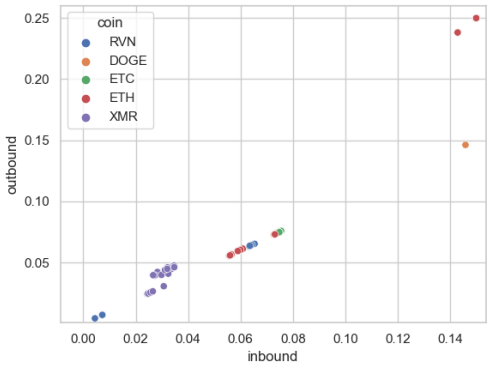
Lastly, the flows with the same inverse of the flow 5-tuple are paired up where source address and port of a flow are the same as the destination address and port of another flow and



(a) Bits/Seconds



(b) Bits/Packets



(c) Packets/Seconds

Fig. 3. Flow statistics for different cryptocurrencies

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.

vice versa. This method produced the plots as shown in Figure 2 and Figure 3.

E. Classification Models

Testing and comparing machine learning models were conducted using the data obtained after feature engineering. The most efficient classifier was selected by comparing the various models' performance. The models used 80% of the dataset for training, and 20% was used for testing. Using the training set, the models used K-fold cross validation with ten folds to give a more well-rounded view of the performance. The machine learning algorithms that have been investigated include K-Nearest Neighbours (KNN), Support Vector Machines (SVM), Decision Trees (DT) and Random Forest Trees (RFT).

IV. Analysis of Results

This section highlights the analyse of the results achieved in this paper.

A. Cryptomining Traffic

This experiment aimed to test whether it could distinguish between cryptomining and normal traffic. There are only two classes considered, which are cryptomining and normal traffic. There were 157 cryptomining flows amongst 84,187 normal traffic flows. The dataset was classified with the models described in Section III-E.

The models are tested over all of the mean flow analysis statistics described in Section III-D. Table I provides the F1 scores of the machine learning models across the different plots.

The results show that overall the classifiers did perform well in distinguishing between cryptomining and normal traffic across the different mean flow statistic averages on the training validation set. The F1 scores average ranged between 82.27% to 98.93% with an average score of 94.62% on the bits per packet graph. Bits per second and packets per second also seemed to perform well across the models, but the SVM result was 49.99%. The SVM's result was understandable as those two particular plots seen in Figure 2(a) and III-D were noisy and thus did not have clear boundaries that were separable.

The models performed well with the training-validation sets, but when put against the test sets, they did not perform well. The overall best performer was the RFT. The AUC scores of the plots seen in Figure 2(a) and III-D were both 50.00% indicating that the classifiers were randomly guessing. The plot shown in Figure 2(b) obtained an AUC score of 95.45%, indicating that the classifier was indeed robust as it was also able to obtain an F1 score of 95.06%.

TABLE I.
F1 scores of different plots

Model	Packets/second	Bits/packet	Bits/second
SVM	49.955	82.273	49.955
DT	92.354	98.933	97.138
RFT	93.427	98.716	97.143
KNN	93.639	98.563	98.015

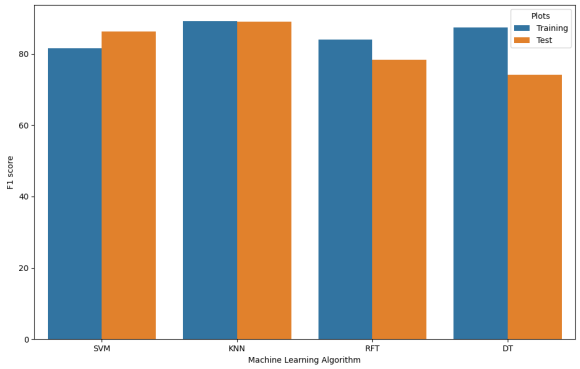


Fig. 4. Machine learning models results from predicting cryptocurrency classes

B. Cryptomining Traffic Between Different Cryptocurrencies

This experiment aimed to see whether the models could classify the different cryptocurrency coins. There were 157 cryptomining flows split across five different cryptocurrency coins. The classifiers were using an 80:20 training test split. The number of flows in the training set was 125, and the classifiers were tested using 32 flows.

Figure 4 shows a graph representing the F1 scores obtained for both training and testing for the machine learning models on the bits per packet plot. As seen, the models performed well overall, with the F1 scores of the training validation set ranging from 81.60% to 89.18% with an average of 85.54%. The test scores were also satisfactory, with F1 scores ranging from 74.18% to 86.30%, averaging 81.97%. This observation shows that the classifiers can generalise well when being tested to classify cryptocurrencies.

The results are different in the cases of the bits per second and packets per second plots, as their training validation F1 scores on average across the models were 96.79% and 83.63%, respectively. However, on the test sets, they achieved average scores of 12.64% and 19.63%. Judging by the disparity of the training validation set and the test sets, this shows that using those two plots causes the models to overfit and thus causes the classifiers to be unable to generalise well. Another cause could be that this was a small dataset; therefore, the classifiers could only be trained on a few cases. The classifiers might be able to perform better if there was a larger dataset.

The bits per packet plot obtained the best test scores, but even then, there were misclassifications within the test dataset. The SVM and KNN classifiers obtained the highest scores with results of 86.30% and 89.06%, respectively. When looking at Figure 3, particularly at Figure III-D, it is understandable why there were misclassifications. The two classifiers mentioned are distance-based algorithms and are affected by noise. Due to the overlapping points as seen in Figure III-D, it was most likely difficult to separate those points. However, it does generalise well as seen, thus making these misclassifications acceptable.

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.

C. Discussion

When it came to this paper, it was evident that the dataset was heavily imbalanced. Therefore, the classifiers might not have been able to detect the intrusion. The first experiment showed that despite the imbalance in the dataset when it came to classifying between crypto-mining and normal traffic, the classifiers that used the bits per packet plot were able to predict the classes successfully.

However, that could be credited to how the dataset was generated. During the creation of the used dataset, only crypto-mining was done on the servers, should there have been other services running on it that displayed a similar relationship in terms of bits per packet. The classifiers might not have been as successful. Therefore, an investigation into using deep learning techniques, such as that used by Zhang *et al.* [7], could help alleviate that problem as deep learning models can find more complex and less obvious relationships between features and can perform better when there is noise.

Overall all the classifiers based on bits per packet have performed well in detecting crypto-mining (and by extension, cryptojacking) with data obtained from a cloud environment. The experiments show that using these machine learning models a cloud environment cryptojacking detection system could be built.

V. Conclusion and Future Work

Cryptojacking is the cybercrime in which crypto-mining is performed on a device without consent. This crime has been extended to take advantage of cloud infrastructure. To help solve this problem, machine learning models were investigated in this paper. This resulted in a model that can classify cryptojacking within a cloud environment, solely analysing network traffic.

For future work, to further validate this model, it could be tested in a real cloud environment and an investigation into making the model robust enough to work under different network conditions, and datasets could be done. Another development could be to build a system that can detect cryptojacking in real time. The exploration of different features to use in detecting cryptojacking using network traffic could prove beneficial. An investigation into the use of deep learning techniques could also be done to potentially improve the model.

Acknowledgements

The authors would like to express their gratitude for the financial support received from Rhodes University. The authors acknowledge that opinions, findings, conclusions and recommendations are those of the authors and that none of the aforementioned sponsors accept liability whatsoever in this regard.

References

[1] W. Bian, W. Meng, and M. Zhang. “Minethrottle: Defending against wasm in-browser cryptojacking”. In: *Proceedings of The Web Conference 2020*. 2020, pp. 3112–3118.

[2] M. Musch, C. Wressnegger, M. Johns, and K. Rieck. “Thieves in the browser: Web-based cryptojacking in the wild”. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 2019, pp. 1–10.

[3] Y. Feng, D. Sisodia, and J. Li. “Poster: Content-agnostic identification of cryptojacking in network traffic”. In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. 2020, pp. 907–909.

[4] K. Jayasinghe and G. Poravi. “A survey of attack instances of cryptojacking targeting cloud infrastructure”. In: *Proceedings of the 2020 2nd Asia pacific information technology conference*. 2020, pp. 100–107.

[5] G. Hong, Z. Yang, S. Yang, L. Zhang, Y. Nan, Z. Zhang, et al. “How you get shot in the back: A systematical study about cryptojacking in the real world”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1701–1713.

[6] I. Petrov, L. Invernizzi, and E. Bursztein. “Coinpolice: Detecting hidden cryptojacking attacks with neural networks”. In: *arXiv preprint arXiv:2006.10861* (2020).

[7] S. Zhang, Z. Wang, J. Yang, X. Cheng, X. Ma, H. Zhang, et al. “MineHunter: A Practical Cryptomining Traffic Detection Algorithm Based on Time Series Tracking”. In: *Annual Computer Security Applications Conference*. 2021, pp. 1051–1063.

[8] X. Ren, H. Gu, and W. Wei. “Tree-RNN: Tree structural recurrent neural network for network traffic classification”. In: *Expert Systems with Applications* 167 (2021), p. 114363.

[9] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. “Network traffic classifier with convolutional and recurrent neural networks for Internet of Things”. In: *IEEE access* 5 (2017), pp. 18042–18050.

[10] K. Patel and R. Srivastava. “Classification of cloud data using bayesian classification”. In: *Int. J. Sci. Res* 2.6 (2013), pp. 2–7.

[11] R. K. Konoth, E. Vineti, V. Moonsamy, M. Lindorfer, C. Kruegel, H. Bos, et al. “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*. 2018, pp. 1714–1730.

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

[13] H. N. C. Neto, M. A. Lopez, N. C. Fernandes, and D. M. Mattos. “Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking”. In: *Annals of Telecommunications* 75.3 (2020), pp. 121–131.

[14] F. Naseem, A. Aris, L. Babun, E. Tekiner, and S. Uluagac. “MINOS: A lightweight real-time cryptojacking detection system”. In: *28th Annual Network and Distributed System Security Symposium, NDSS*. 2021.

[15] D. Schneider. “A faster fast fourier transform”. In: *IEEE Spectrum* 49.3 (2012), pp. 12–13.

[16] J. Z. i Muñoz, J. Suárez-Varela, and P. Barlet-Ros. “Detecting cryptocurrency miners with NetFlow/IPFIX network measurements”. In: *2019 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE. 2019, pp. 1–6.

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 04,2024 at 07:28:55 UTC from IEEE Xplore. Restrictions apply.