2022 International Conference on Intelligent and Innovative Computing Applications
https://doi.org/10.59200/ICONIC.2022.006

Cryptojacking Malware Detection in Docker Images Using Supervised Machine Learning

Saide Manuel Saide
Faculty of Engineering
Lurio University
Pemba, Mozambique
saide.saide@unilurio.ac.mz

Ednilson Luis Alfredo Sarmento
Faculty of Engineering
Lurio University
Pemba, Mozambique
esarmento@unilurio.ac.mz

Felermino D. M. A. Ali
Faculty of Engineering
Lurio University
Pemba, Mozambique
felermino.ali@unilurio.ac.mz

Abstract
Nowadays, Docker Containers are currently being adopted as industry standards for software delivery, because they provide quick and responsive delivery and handle performance and scalability challenges. However, attackers are exploiting them to introduce malicious instructions in publicly available images to perform unauthorized use of third-party's computer resources for Cryptojacking. We developed a machine learning based model to detect Docker images that lead to cryptojacking. The dataset used is composed of 800 Docker images collected from Docker hub, half of which contains instructions for cryptomining, and the other half does not contain such instructions. We trained 10 classification algorithms and evaluated them using the K-Fold Cross Validation approach. The results showed accuracy scores ranging from 89% to 97%. Stochastic Gradient Descent for Logistic Regression outperformed the other algorithms reaching an accuracy score of 97%. With these results, we conclude that machine learning algorithms can detect Docker images carrying cryptojacking malware with a good performance.

Keywords: Cryptojacking, Docker images, Machine Learning, Cryptomining, Cybersecurity

1. Introduction

Docker containers are virtualization technology based on the operating system level, where applications and all their dependencies and source code are packaged in a single resource called image (Karn et al., 2021), from which multiple containers can be initiated. Containers provide huge advantages when managing a complex infrastructure (Şengül et al., 2021), and because of that, their use is on the rise. However, cybercriminals found an opportunity to exploit them to introduce malicious activities such as attaching instructions in Docker images to trigger unauthorized activities. For instance, due to the popularity and increasing value of cryptocurrency, malicious cryptomining tends to be a common attack on docker images (Carlin et al., 2020). This type of attack is also known as Cryptojacking, which is defined as the unauthorized use of computational resources to mine cryptocurrency (Carlin et al., 2020).

From a cybersecurity point of view, malicious cryptomining is a threat. It affects the victims not only by illegally using and wasting their CPU resources but also by causing financial damages due to extra electricity costs (Hong et al., 2018; Tekiner et al., 2021). Cryptojacking attacks can be carried out by different attack vectors, such as websites, operating systems, Random Access Memory - RAM (Varlioglu et al., 2022), and docker images (Liu et al., 2020; Karn et al., 2021). We focus on the application of supervised machine learning approach to detect cryptojacking in docker images. We collected 800 Docker images from Docker hub – a public repository provided by Docker, where 400 images had instructions for cryptojacking. The remainining 400 images were free of malicious instructions. We trained and evaluated 10 classification algorithms, using K-Fold Cross Validation sampling, and found that Stochastic Gradient Descent for Logistic Regression had the highest accuracy scores (97%), and K-Nearest Neighbors (KNN) algorithm had the lowest accuracy scores (89%).

The paper is structured as follows: Section 2 presents research done by other authors related to criptojacking attack mitigation, while the methodology used to carry out this research: data set collection, data set preprocessing, data set transformation, and model evaluation is described in Section 3. Section 4 describes the experiments:

training and testing aspects. This section also presents the results obtained and the discussion. The paper conclusions are presented in Section 6.

2. Related Works

Compared to hypervisor-based virtualization, containers showed to be lightweight systems that rationalizes hardware usage, provide performance nearly equal to non-virtualized systems. Because of that, they are being widely adopted (Kwon & Lee, 2020). Nevertheless, Docker containers are known to be vulnerable (Wenhao & Zheng, 2020; Huang et al., 2019), as they share operating system kernels, and run on root user by default. This fact allows containers to easily execute privileged operations, as well as easily access hardware resources. This way, computer attacks, such as criptojacking, can easily succeed via containers.

With the proliferation of cryptojacking attacks (Yulianto et al., 2019), especially through docker containers (Karn et al., 2021; Liu et al., 2020) cryptojacking defense has become on high demand. Different detection mechanisms have been proposed in literature ever since. For instance, Lachtar et al. (2020) worked on a solution that relies on tracking instruction executed within CPUs, an approach that can detect cryptojacking attack regardless of vector application type. Tanana (2020) researched cryptojacking detection by monitoring CPU loads caused by application software; his work focused on web pages and executable cryptojacking types. They achieved a success rate of 82% , and was designed by following the decision tree algorithm. Hong et al. (2018) developed a cryptojaking detection and prevention tool named CMTracker. For this work, they collected 853,936 pages as dataset on the web, and were able to detect 2,770 scripts for cryptojacking.

The literature also shows an increasing interest to use machine learning approaches as they allow computers the ability to automatically learn from its experience by feeding them with the appropriate data while identifying patterns to later make predictions from related unseen data with minimal human intervention. For example, Wang et al. (2018) have proposed a solution named SEISMIC, for interrupting cryptojaking based on web browsers. The proposed method relies on semantic signature-matching, by monitoring application software behavior instead of syntax analysis, which makes it more robust to evade code obfuscation techniques. They used supervised machine learning approach, and tested Support Vector Machine classifier, and achieved a F1 score greater or equal to 98%. Carlin et al. (2018), worked on supervised Machine Learning for cryptojacking detection and experimented with Random Forest classifier; they aimed to dynamically analyse opcode on non-executable files. The results showed scores achieving an accuracy of 100% on the test dataset. They created four different datasets namely: cryptomining (with 296 data points), deactivated cryptiminig (containing 194 samples), canonical (with 359 observations), and canonical injected (57 samples). Saad et al. (2019) proposed a solution that uses an unsupervised machine learning approach to mitigate the cryptojacking attacks via browsers. They collected dataset published from Pixalate and Netlab 360. They collected a total of 5,703 observations. They worked with Fuzzy C-Means (FCM) clustering algorithm to group the observations in categories such as, malicious, and benign, and they obtained an accuracy of 96.4% .

Tahir et al. (2019) proposed a real-time cryptojacking detection on browser code based on hardware-assisted profiling using supervised machine learning approach. They collected 590 data points for their dataset by recording Hardware Performance Counters values and used Random Forest classifier. They achieved an accuracy score of 99.35%. Nukala (2020) worked with machine learning to detect website based cryptojacking based on CPU activity data; they experimented with five classification algorithms while collecting cache hit and cache misses information, and achieved an accuracy of 96.25%. Lachtar el al. (2021) presented a solution for dynamically criptojacking attacks detection, along with other techniques. They used supervised machine learning approach and evaluated five classification algorithms, the dataset was composed of 272 observations composed of CPU instructions, and achieved an accuracy score result of 100% on the detection of cryptojacking for Zcash and Monero cryptocurrencies. Karn et al. (2021) developed a research work where they designed a framework based on supervised machine learning approach to detect containers that launch cryptomining activity, by collecting legitimate pods syscalls along with cryptomining hijacked pods syscalls as dataset. They tested four classifiers and obtained 97.1% as the highest accuracy score.

ISBN: 978-99949-0-888-2

Hu et al. (2021) worked on a solution to cryptojacking detection based on network traffic using supervised machine learning approach, and experimented five classification algorithms. Their result on this research achieved an accuracy score of 99.91%. Their dataset was composed of captured network traffic, which contained cryptojacking traffic for bitcoin currency. They collected 400,000 samples and added USTC-TFC2016 dataset to their dataset. Wang et al. (2021) presented a solution called MineDetector, they used supervised machine learning to detect web browser-side criptojacking scripts. They constructed a dataset containing features from javaScript source code. They had two datasets, one with 130 cryptomining JavaScripts, obtained from virusshare.com and Wayback machine, and another with 2,668 benign JavaScripts, obtained from BootCDN. They experimented with five classification algorithms and achieved an accuracy score of 99.41%. Gomes and Correia (2020) worked on cryptojaking detection based on the machine learning approach – they evaluated and tested various classification algorithms, and obtained an average precision score of 92.5%. The research focused on web page CPU usage monitoring. They obtained the training dataset by crawling web pages.

Our research differs from the works reported in the literature because it is based on detecting cryptojaking instructions directly from Docker images, i.e. before the instantiation of containers, while most of the related works we found were focused on detecting fileless, scripting, and operating system based cryptojacking attacks by monitoring CPU load, opcode, etc. With exception of one research that focused on network traffic monitoring, and another one that focused on monitoring pods system calls. Although all the related work used its own dataset, most of the datasets were collected from CPU behaviour, and almost all studies used the same classification algorithms: Naive Bayes, Support Vector Machines - SVM, K-Nearest Neighbor - KNN, Logistic Regression and some related work used Random Forests, and Decision Trees.

3. Methodology

3.1. Data set

We installed a VMware Workstation 16 Pro Virtual Machine (VM), with Ubuntu 18.04 LTS operating system where we installed Docker 20.10.7. We then composed a bash script instructed with the “docker pull” command, and we were able to download Docker images from Docker hub to our local VM 800. The dataset was balanced between two classes (i.e., cryptojacking, not cryptojacking) each with 400 cryptojacking examples. Subsequently, in our local VM we used the “docker history” command for each Docker image to extract all the image instructions into a plain text file. We obtained 800 different text files. Figure 1 illustrates four observations from the resulting text file, which contained 800 observations in total. A sample of the dataset is depicted in Table 1 (the ellipse inside the square brackets means there is more text).

Table 1: Illustration of subset of the resulting dataset file after joining all the image text files into one text file

image	class
ENTRYPOINT [“. /xmrig"] nop [...]	1
CMD ["bash"] nop […]	0
nop CMD ["-origin" "url" "-user" …]	1
nop WORKDIR /home/user […]	0

3.2. Data Preprocessing

Data preprocessing was carried out as an initial step prior to training. We preprocess the data using Jupyter Notebook with Python 3.8 scripting language. We followed the consecutive steps listed below:

- Lowercasing - casing can be problematic when it comes to text data. For instance, the words/tokens “copy” and “COPY” can be recognized as different words. So, to avoid this problem and to minimize the number of features in the dataset we applied lowercasing to the text.
- Noise cleaning - we identified URLs in each image. Here we kept unique URLs to avoid too many URLs which could increase the number of features in the dataset. Thus, using regex we substituted all the URLs in the text by the word “*URL”*. The regex used for matching URLs was: r‘http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+’;
- Removal of punctuations - Similarly, we removed all the punctuations and special

ISBN: 978-99949-0-888-2

characters using regex instructions by matching the following pattern: r‘[,.\"!@#$%^*(){}?;`~:<>_]’;

- Tokenization - we used regex expressions to split the text into small units/tokens. These tokens were then used as features to feed the algorithms during training. The regex rule used for the tokenization was: ‘r[\w’]+|[""!"#$%&’()*+,-./:;<=>?@[\]^_‘{|}~""\\]’;

3.3. Data Transformation

To perform the train and test, we first needed to transform the text dataset to numerical representation. For that, we experimented with two approaches: Bag of Words (Qader et al., 2019) and Term Frequency-Inverse Document Frequency - TF-IDF (Qaiser & Ali, 2018).

Bag of Words allows the representation of text data in fixed-length numeric vectors. We experimented with n-gram models to represent the features of the data set; however, uni-gram was chosen to represent the features, since it was showed to provide better results. TF-IDF on the other hand, is a statistical approach that measures the relevance of each word in a textual data. Table 2 shows one record from the resulting dataset after transformation using the Bag of Words approach; for the sake of space management, we only display one data point. Table 3 illustrates the same subset features of the same data point after applying the TF-IDF approach to transform the dataset.

Table 2: Bag of Words representation of three features of one data point from the dataset

xmrig	cpu	miner	…
4	1	3	…

Table 3: TF-IDF representation of three features of one data point from the dataset

word	TF	IDF	TF-IDF
xmrig	4	2.387544	0.091413
cpu	1	3.466353	0.033179
miner	3	3.279142	0.094162
…	…	…	….

3.4. Model evaluation

The performance of the models was measured using classification accuracy. The evaluation of the model was conducted in two phases: firstly, we trained and tested the algorithms, and we stored both training and testing scores. Then, we compared the two scores (training and testing scores) for each algorithm, for overfitting and underfitting analysis. Along with the testing accuracy, we used confusion matrices to better analyze the number of false positives and false negatives for each algorithm, to better choose the algorithm that performed better. Figure 1 illustrates the system diagram that shows the method of detecting the cryptojacking malware.

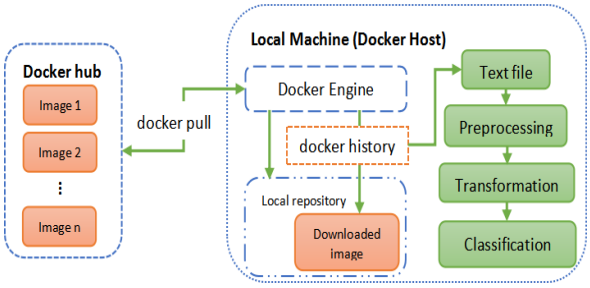

Figure 1: System diagram showing the cryptojacking malware detection method

4. Experiments and Results

4.1 Experiments

Our experiments were carried out using Jupyter Notebook. After transforming the dataset, we split it in two, 80% for training, 20% for test. Then, using the K-Fold Cross Validation sampling approach (with k = 5), we submitted the training dataset (80%) to 10 classification algorithms. From the 10 algorithms 6 were the same algorithms used by the related work, and we added 4 more algorithms namely: Stochastic Gradient Descent for Logistic Regression - SGD (LR), Stochastic Gradient Descent for SVM - SGD (SVM), Gradient Boosting Classifier (GBoost), and Adaptive Boost Classifier (AdaBoost). These algorithms were added because they are known by the literature to perform well on binary classification problems, which is the case of our research task.

To automate the experiment workflow, we used grid-search, and pipelines. Grid-search was used to tune algorithms parameters, to tune the appropriate number of folds to K-Fold Cross Validation, as well as to find the better approach between Bag of Words and TF-IDF, before feeding the data to the algorithms, while pipelines were used to orchestrate the experiments sequence operations during the algorithms training.

4.2. Results

The training results are illustrated in Table 4. The second column (*Acc*) represents the accuracy score for each algorithm, it is the arithmetic average of the five folds performed by K-Fold

Cross Validation. The third column (*std*) represents the standard deviation of the respective accuracy, and the fourth column (*CI*) represents the confidence interval in which the accuracy falls, with a confidence of 95%.

Table 4: Training results for each classifier

Algorithm	***Acc***	***Std***	***CI***
LR	0.928	0.026	[0.896; 0.960]
SGD (LR)	0.948	0.021	[0.922; 0.975]
SVM	0.948	0.014	[0.930; 0.966]
SGD (SVM)	0.950	0.021	[0.924; 0.976]
NB	0.953	0.023	[0.924; 0.982]
KNN	0.917	0.021	[0.891; 0.943]
RF	0.941	0.020	[0.916; 0.966]
DT	0.933	0.011	[0.919; 0.946]
GBoost	0.948	0.009	[0.937; 0.960]
AdaBoost	0.958	0.020	[0.913; 0.962]

Table 5 shows the testing results, the accuracy (*Acc*) column is the main evaluation metric in this research, it is presented with other evaluation metrics, namely: precision (*Prec*), recall (*Rec*), 1-Measure (*F1-M*).

Table 5: Testing results for each classifier

Algorithm	***Acc***	***Prec***	***Rec***	***F1-M***
LR	0.912	0.946	0.875	0.909
SGD (LR)	0.969	1.000	0.938	0.968
SVM	0.963	0.987	0.938	0.962
SGD (SVM)	0.950	0.986	0.912	0.948
NB	0.944	0.986	0.900	0.941
KNN	0.894	0.920	0.863	0.890
RF	0.950	0.986	0.912	0.948
DT	0.938	0.986	0.887	0.934
GBoost	0.944	1.000	0.887	0.940
AdaBoost	0.925	0.925	0.925	0.925

We used confusion matrices along with the testing accuracy to better judge the results. Our intention was to see how balanced the classification task was, by analysing the amount of well classified and misclassified data points. Figure 2 illustrates the confusion matrix for the SGD (LR) algorithm whereas, Figure 3 shows the confusion matrix of the SVM. SGD (LR) achieved the best accuracy score between them.

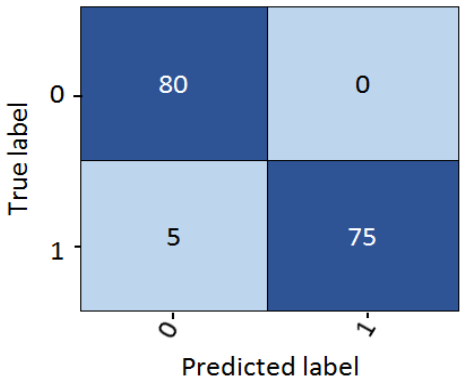

Figure 2: Confusion matrix of Stochastic Gradient Descent for Logistic Regression classification results

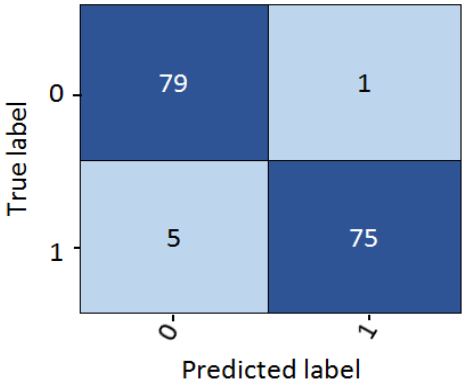

Figure 3: Confusion matrix of Support Vector Machine classification results

For the two confusion matrices, the values in the main diagonal (the dark blue squares), express the values that were well classified, while the values in the secondary diagonal (the light blue squares) represent the misclassified values (the false positives and the false negatives).

4.3. Discussion

The testing results (Table 5) illustrates that SGD (LR) achieved the best result – it achieved 96.9% accuracy, while SVM achieved 96.3% accuracy. These results are close to each other, however, observing the respective confusion matrices (Figure 2 and Figure 3), we find that SGD (LR) had 5 false negatives in its classification, while it did not have any false positives, while SVM had 1 false positive, making this algorithm less accurate than SGD (LR). On the other hand, we observed that all the obtained results fell within the calculated confidence interval. We also observed that the accuracy of GBoost in the training phase, obtained the least standard deviation (0.009), meaning that the algorithm had the least dispersion among the various accuracies. Table 4 and Table 5 present the results achieved

by using the TF-IDF model, which produced better results in the experiments. This can be justified by the presence of a considerable amount of noise in the dataset, so it needed to define the importance of the terms.

Since containers are commonly adopted in cloud data centres, our research can minimize the unauthorized use of the data centre's hardware resources by cryptojacking attackers. Our proposal is illustrated in Figure 4 — a system diagram showing how to minimize the unauthorized use of cloud data centre's hardware resources. The cloud data centre pulls the Docker images from Docker hub to a server within the data centre. After that, the downloaded images are submitted to the verification process to ensure that the images do not contain cryptojacking instructions. If the safety of the image is verified, the image is stored in the local cloud data center repository where the data centre's tenants can transparently request to instantiate their containers.

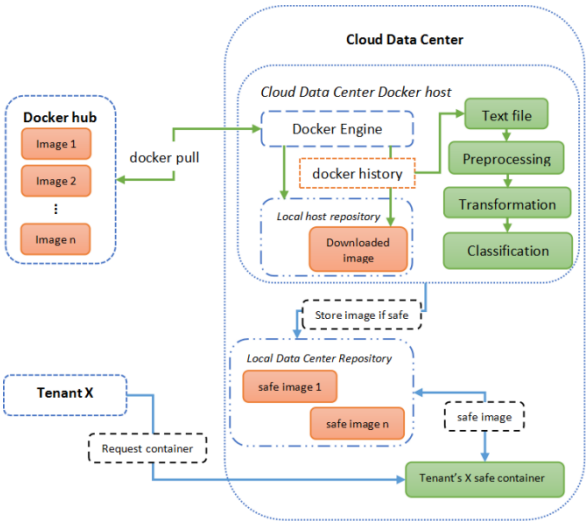

Figure 4: System diagram showing how to minimize the cryptojaking on cloud data centers

6. Conclusion

This research aimed to develop a predictive model based on Machine Learning to detect cryptojacking malware in Docker images. To achieve this goal, we first built a dataset using the "docker history" command for Docker images collected from Docker Hub. We pre-processed and transformed the dataset and performed several experiments using supervised Machine Learning classification algorithms. The experiments showed that the Stochastic Gradient Descent algorithm for Logistic Regression SGD (LR) outperformed the other algorithms in terms of performance, achieving an accuracy score of nearly 97%. With a confidence of 95% it can be said that the highest average accuracy for SGD (LR) is between 92.2% and 97.5%. With these results, we conclude that Machine Learning classification algorithms can detect Docker images carrying cryptojacking malware with an acceptable performance (accuracy). The study demonstrated that the "docker history" command can serve as a basis to feed a cryptojacking malware detection system. It also demonstrated that different Machine Learning algorithms can be used to build a predictive model for detecting cryptojacking malware in Docker images.

7. References

Carlin, D., O'Kane, P., Sezer, S., & Burgess, J. (2018). Detecting cryptomining using dynamic analysis. *16th Annual Conference on Privacy, Security and Trust (PST)*, 1-6. https://doi.org/10.1109/PST.2018.8514167

Carlin, J. Burgess, P. O'Kane & Sezer, S. (2020). You could be mine(d): The rise of cryptojacking. *in IEEE Security & Privacy*, *18*(2), 16-22. https://doi.org/10.1109/MSEC.2019.2920585

Gomes, F. & Correia, M. (2020). Cryptojacking detection with cpu usage metrics. *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, 1-10. https://doi.org/10.1109/NCA51143.2020.9306696

Hong, G., Yang, Z., Yang, S., Zhang, L., Nan, Y., Zhang, Z., Yang, M., Zhang, Y., Qian, Z., & Duan, H. (2018). How you get shot in the back: A systematical study about cryptojacking in the real world. *In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, 1701-1713. https://doi.org/10.1145/3243734.3243840

Hu, X., Shu, Z., Song, X., Cheng, G., & Gong, J. (2021). Detecting cryptojacking traffic based on network behavior features. *2021 IEEE Global Communications Conference (GLOBECOM)*, 01-06. https://doi.org/10.1109/GLOBECOM46510.2021.9685085

Huang, D., Cui, H., Wen, S., & Huang, C. (2019). Security analysis and threats detection techniques on docker container. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 1214-1220.

https://doi.org/10.1109/ICCC47050.2019.9064441

Karn, R. R., Kudva, P., Huang, H., Suneja, S., & Elfadel, I. M. (2021). Cryptomining detection in container clouds using system calls and explainable machine learning. *in IEEE Transactions on Parallel and Distributed Systems*, *32*(3), 674-691. https://doi.org/10.1109/TPDS.2020.3029088

Kwon, S., & Lee, J. -H. (2020). DIVDS: Docker image vulnerability diagnostic system. *in IEEE Access*, *8*, 42666-42673. https://doi.org/10.1109/ACCESS.2020.2976874

Lachtar, N., Elkhail, A. A., Bacha, A. & Malik, H. (2021). An application agnostic defense against the dark arts of cryptojacking. *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 314-325. https://doi.org/10.1109/DSN48987.2021.00044

Lachtar, N., Elkhail, A. A., Bacha, A., & Malik, H. (2020). A cross-stack approach towards defending against cryptojacking. *in IEEE Computer Architecture Letters*, *19*(2), 126-129. https://doi.org/10.1109/LCA.2020.3017457

Liu, P., Ji, S., Fu, L., Lu, K., Zhang, X., Lee, W., Lu, T., Chen, W., & Beyah, R. (2020). Understanding the security risks of docker hub. *In Computer Security – ESORICS 2020: 25th European Symposium on Research in Computer Security*, 257–276. https://doi.org/10.1007/978-3-030-58951-6_13

Nukala, V. S. K. A. (2020). Website cryptojacking detection using machine learning: Ieee cns 20 poster. *2020 IEEE Conference on Communications and Network Security (CNS)*, 1-2. https://doi.org/10.1109/CNS48642.2020.9162342

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An overview of bag of words; importance, implementation, applications, and challenges. *2019 International Engineering Conference (IEC)*, 200-204. https://doi.org/10.1109/IEC47844.2019.8950616

Qaiser, S., & Ali, R. (2018). Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, *181*. https://doi.org/10.5120/ijca2018917395

Saad, M., Khormali A., & Mohaisen, A. (2019). Dine and dash: Static, dynamic, and economic analysis of in-browser cryptojacking. *APWG Symposium on Electronic Crime Research (eCrime)*, 1-12. https://doi.org/10.1109/eCrime47957.2019.9037576

Şengül, Ö., Özkılıçaslan, H., Arda, E., Yavanoğlu, U., Doğru İ. A., & Selçuk, A. A. (2021). Implementing a method for docker image security. *2021 International Conference on Information Security and Cryptology (ISCTURKEY)*, 34-39. https://doi.org/10.1109/ISCTURKEY53027.2021.9654383

Tahir, R., Durrani, S., Ahmed, F., Saeed, H., Zaffar, F., & Ilyas, S. (2019). The browsers strike back: Countering cryptojacking and parasitic miners on the web. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 703-711. https://doi.org/10.1109/INFOCOM.2019.8737360

Tanana, D. (2020). Behavior-based detection of cryptojacking malware. *Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, 0543-0545. https://doi.org/10.1109/USBEREIT48449.2020.9117732

Tekiner, E., Acar, A., Uluagac, A. S., Kirda, E., & Selcuk, A. A., (2021). Sok: Cryptojacking malware. *IEEE European Symposium on Security and Privacy (EuroS&P)*, 120-139. https://doi.org/10.1109/EuroSP51992.2021.00019

Varlioglu, S., Elsayed, N., ElSayed Z., & Ozer, M. (2022). The dangerous combo: Fileless malware and cryptojacking. *SoutheastCon 2022*, 125-132. https://doi.org/10.1109/SoutheastCon48659.2022.9764043

Wang, P., Sun, Y., Huang, C., Du, Y., Liang, G., & Long, G. (2021). Minedetector: Javascript browser-side cryptomining detection using static methods. *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*, 87-93. http://doi.org/10.1109/CSE53436.2021.00022

Wang, W., Ferrell, B., Xu, X., Hamlen, K., & Hao, S. (2018). Seismic: Secure in-lined script monitors for interrupting cryptojacks.

ESORICS, 122-142. https://doi.org/10.1007/978-3-319-98989-1_7

Wenhao, J., & Zheng, L. (2020). Vulnerability analysis and security research of docker container. *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 354-357. https://doi.org/10.1109/ICISCAE51034.2020.9236837

Yulianto, A. D., Sukarno, P., Warrdana, A. A., & Makky, M. A., (2019). Mitigation of cryptojacking attacks using taint analysis. *4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 234-238. https://doi.org/10.1109/ICITISEE48480.2019.9003742