NoCrypto: A Web Mining Behavior Detection Method Based on RGB Images

Hui Wang[1], Yu Zhang[1], Xiaoming Pan[2(✉)], and Weiyi Huang[1]

[1] Guangdong Polytechnic Normal University, GuangZhou 510665, GuangDong, China
bullzhangyu@gpnu.edu.cn

[2] Zhejiang Electronic Information Products Inspection and Research Institute (Key Laboratory of Information Security of Zhejiang Province), HangZhou 310007, ZheJiang, China
pxm@zdjy.org.cn

Abstract. In recent years, there has been a growing prevalence of mining web pages using the new web technology of WebAssembly (WASM), resulting in the unauthorized exploitation of user resources. However, existing detection methods have shown limited ability to counter obfuscation techniques and have exhibited low detection efficiency. To address these issues, this paper proposes a novel static detection method based on the visualization of WASM modules. The proposed method involves instantiating the binary files of the WASM mining operations within web pages. These binary files are then combined with the information of local entropy and global entropy, resulting in the visualization of RGB images. Compared to grayscale images, RGB images retain more of the original file information. After training and learning the image features using a convolutional neural network (CNN), the model achieves an impressive accuracy rate of 99.18% when tested on real-world web pages. This accuracy is approximately 2% higher than that of existing visualization-based detection methods. Moreover, the model exhibits a shorter execution time. The proposed NoCrypto method demonstrates quick execution speed and accurate detection.

Keywords: Web mining · WebAssembly (WASM) · RGB visualization detection · CNN model

1 Introduction

Since the introduction of WebAssembly (WASM) technology in March 2017 [1], it has gained attention due to its capabilities as a low-level bytecode language that enables faster transmission, parsing, and execution of programs. Unfortunately, hackers have also taken notice of these advantages. They have begun utilizing this new web technology to mine memory-constrained cryptocurrencies, such as Monero. This new form of mining reached its peak by the end of 2017.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2025
Published by Springer Nature Switzerland AG 2025. All Rights Reserved
J. Cai et al. (Eds.): SPNCE 2023, LNICST 525, pp. 224–236, 2025.
https://doi.org/10.1007/978-3-031-73699-5_16

According to a research paper [2], it was found that among the top one million websites ranked by Alexa, one out of every 600 sites used WebAssembly. Furthermore, one-third of these sites spent more than 75% of their time executing WebAssembly instead of JavaScript code. Mining websites accounted for 32% of the sites using WebAssembly. Consequently, this shift has prompted a transition in mining detectors from focusing on JavaScript to addressing WebAssembly.

In order to detect mining websites based on WebAssembly (WASM) technology and protect computer users' computational resources from unauthorized usage, researchers from both domestic and international backgrounds have conducted studies on web mining behaviors, yielding significant achievements [3]. Existing detection techniques can be mainly categorized into three types: rule-based detection, machine learning-based detection, and deep learning-based detection.

Rule-based detection techniques for WebAssembly-based mining websites involve extracting the WASM modules used for mining and converting them into WAT (WebAssembly Text) format. From the WAT format, the bytecode or opcode instructions are extracted and matched against semantic signatures based on the counts of these instructions. This approach aims to detect mining activities based on the semantics of WASM bytecode or opcode instructions. However, the effectiveness of this detection technique is limited when faced with new generation mining viruses that employ common obfuscation techniques such as code obfuscation, one-click encryption, throttling evasion, and URL obfuscation. These techniques make it challenging for the rule-based detection method to deobfuscate the mining scripts effectively. Additionally, the bytecode and opcode of new mining scripts may differ from the previously defined instructions, rendering this detection method ineffective against obfuscated mining scripts and new types of mining scripts.

Machine learning based detection technique by applying the principle that the WASM module will be called multiple times during the execution of the mining website, the number of Wasm modules in the stack function is used as a numerical feature to identify the mining behavior, and the detection is carried out by fusing multiple features through the method of machine learning, which requires a certain amount of running overhead due to the use of dynamic analysis.

In response to the above mentioned problems of poor anti-obfuscation ability, low recognition rate of new mining scripts, and dynamic analysis requiring some running overhead, deep learning methods are beginning to be used. Deep learning based detection techniques combine deep learning with visualization methods to convert the binary file form of the WASM module into an image, and then use the deep learning model to identify it, this method has a strong anti-obfuscation ability, has a high recognition rate of all kinds of mining scripts, and belongs to the static analysis, which doesn't require a great running overhead. However, the existing deep learning detection methods convert WASM into grayscale images through the B2M grayscaling algorithm, and the information enriched in the grayscale image and the clarity of the image contour are not as good as the RGB image, and the detection accuracy is still to be improved.

The RGB visualization method proposed in this paper generates images with clearer contours, images generated by different mining scripts are more similar, the degree of differentiation between them and non-mining scripts is greater, and the final detection effect is better than the grayscale method.

This paper's detection method offers several advantages compared to existing techniques:

- In terms of image visual analysis, the proposed method introduces a novel approach for visualizing binary files as RGB images. By combining local entropy and global entropy, it effectively retains a greater amount of local and global feature information. This technique enhances the representation of the binary files, allowing for more comprehensive analysis and detection of mining behaviors. The utilization of both local and global entropy ensures that important characteristics of the files are preserved in the resulting RGB images.
- This paper presents a method for visualizing mining-related WebAssembly (WASM) features to achieve mining web page detection. Compared to existing grayscale-based methods for WASM feature extraction, the proposed method provides clearer image textures. Furthermore, the visual distinction between benign and malicious images is more prominent in terms of image contours. As a result, the overall performance of the proposed method is superior, offering improved accuracy and reliability in distinguishing between mining and non-mining web pages.
- The proposed method is characterized by its simplicity, as it eliminates the requirement for operators to possess domain-specific knowledge in the field of mining websites. Once the model is trained, the method only requires the input of a website URL for detection. This greatly reduces the analysis complexity, while simultaneously improving the speed and efficiency of the analysis process. As a result, the method offers a user-friendly and time-saving approach for detecting mining web pages.

2 Related Work

This paper focuses on bytecode characterization of WASM files. In the rule set based mining web page detection technique, Konoth [4] et al. converted the WASM module used for mining by extracting it from a website into WAT text format, which contains assembly bytecode. They used a debugger to convert the WASM into linear assembly byte code, followed by semantic signature matching, and finally a machine learning classification algorithm to obtain a 93.78% detection rate. Wang [5] et al. analyzed to reveal the distribution of WASM instructions executed by the mining scripts, which were used at runtime to differentiate between mining and non-mining activities. The semantic signature matching of WASM bytecode instruction counts can detect whether WASM is mining or not with 98% accuracy. This technique performs semantic signature matching after manually creating a rule set by reading bytecode instructions from the WAT text format [6–9]. In machine learning based mining webpage detection techniques,

Outguard [10], MinerAlert [11], etc. by combining the presence of WASM module as one of the mining behavioral characteristics combined with performance analysis of the hybrid detection method, the use of Random Forest detection model ultimately obtains an accuracy rate of 99.59%.

Unlike the above detection methods for WASM assembly byte codes or instruction opcodes, the paper MINOS [12], which was made public in 2021, achieves the combination of benign WASM and malicious WASM visualization and deep learning detection by grayscaling the extracted WASM binaries into images with the help of the deep learning model CNN, which is the beginning of the combination of the field of mining webpage detection with deep learning model combination, finally the method achieved 98.97% detection rate, but its detection rate and the stability of the model still need to be improved.

The main shortcomings of current detection methods are:

- JS script-based detectors fail to detect WASM code;
- Existing WASM detectors are susceptible to obfuscation techniques;
- For the emergence of a new generation of mining scripts based on the WASM module, the existing WASM detectors are ineffective in detecting them.

In order to solve the above problems, based on the premise that the visual representations of different types of mining scripts are similar and differ greatly from those of benign scripts, this paper proposes a new visualization method for mining webpage detection based on the mining features of WASM binaries, i.e., the RGB visualization method that combines the local entropy and global entropy with the purpose of improving the recognition and detection rate of novel mining webpages.

3 NoCrypto

As shown in Fig. 1, the method consists of three parts: collection of WASM samples, sample RGB visualization, and training and detection of image classification models.

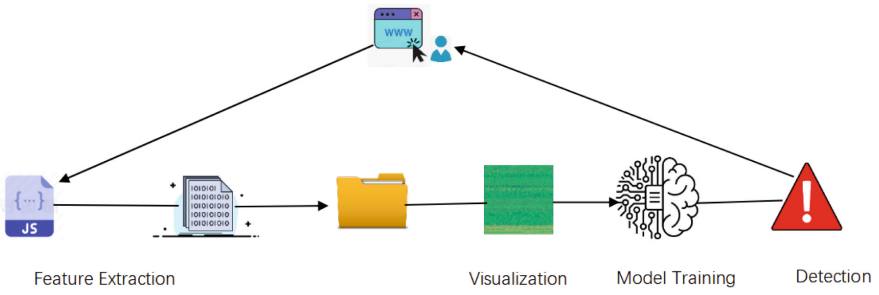


Fig. 1. Design diagram of NoCrypto method flow

3.1 WASM Sample Collection

The sample collection method utilized in this paper draws inspiration from the approach described in [2]. A script written in Node.js is used to automatically collect WASM binary files when users browse the web. These files are then downloaded and stored in a specified folder. The method leverages Puppeteer, a Node.js library that allows communication with and manipulation/control of Google Chrome through an advanced API and DevTools protocol. The code wraps instances of JavaScript functions such as WebAssembly. InstantiateStreaming, and records the binary files of the modules to the Node.js backend.

3.2 WASM Sample RGB Visualization

A WASM module binary consists of a series of hexadecimal numbers. This vector of hexadecimal values can be modified and converted to an image. In such a conversion, the WASM binary is first converted to a vector of 8-bit unsigned integers (UINT8) and then reshaped into a one-dimensional array. This reshaped array is then divided by 255 in order to represent each integer as a pixel with a value ranging from 0 to 255 (0 is black and 255 is white). These pixel values together form the WASM module binary representation of the red channel of the RGB image; the one-dimensional array is transformed into a two-dimensional matrix, and 60 times the local entropy of each value is used as a representation of the green channel of the RGB image, with the local entropy calculated using the specific formula below:

$$H_local = -60 * n/9 * log2(n/9) \tag{1}$$

where n is the frequency of each value in the sub-matrix, the formula represents multiplying the probability of each value in the sub-matrix (n/9) with its corresponding amount of information (-log2(n/9)), and then summing all the values within the sub-matrix to obtain the value of local entropy H_local.

To represent each numerical value of the global entropy as the blue channel in the RGB image, we multiply the value by 60. The calculation process of the global entropy is as follows:

$$L_local = -60 * n/N * log2(n/N) \tag{2}$$

where n is the frequency of each value in the two-dimensional matrix and the total number of values in the two-dimensional matrix is N. The formula represents multiplying the probability of each value in the two-dimensional matrix (n/N) with its corresponding amount of information (-log2(n/N)), and then summing up all the values within the two-dimensional matrix to obtain the value of global entropy H_local.

The R, G, and B values mentioned above can be mapped to the three channels to form an RGB image. An example of the binary to image conversion for malicious and benign web pages is illustrated in Fig. 2. From Fig. 2, it can

be observed that the binary files used for cryptocurrency mining exhibit striking visual similarity when converted to grayscale images. This similarity arises from their utilization of hashing algorithms to implement proof-of-work (PoW) schemes, resulting in similar syntactic and semantic features. On the other hand, the images representing malicious WASM binary files differ from those associated with benign web pages.

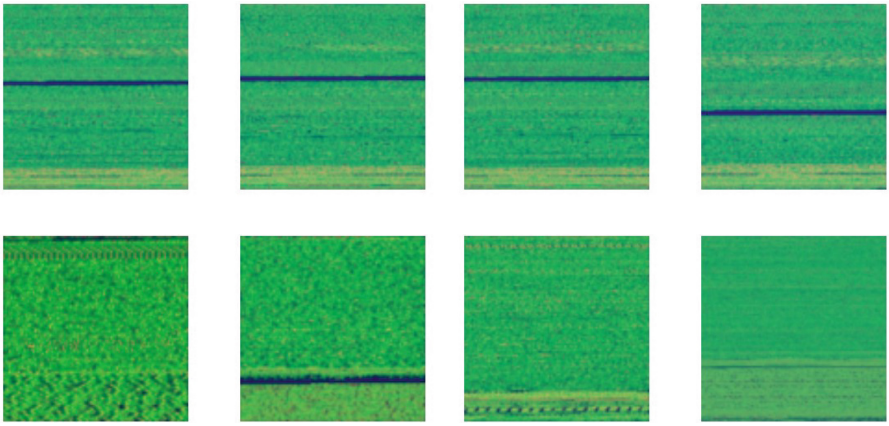


Fig. 2. The first line represents the RGB image obtained by visualizing the WASM binary file in a mining web page. The second line represents the RGB image obtained by visualizing the WASM binary file in a benign web page.

The visualization process is implemented using a recursive function written in python3, the implementation details are described in Algorithm 1. The module first reads all the WASM binaries under the folder, and in lines 2–3 the while loop checks every second if the destination folder of the extracted WASM binaries is empty. In lines 4–7, once the binary file is collected and added to the folder, it is opened and prepared for further preprocessing. In line 5, the folder Wasm_images is declared, which will store the converted images. Line 6 ensures that if a site loads more than one Wasm module, each downloaded module will be accessed in each iteration of the for loop and preprocessed. In lines 8–15, the binary file byte stream is converted into a two-dimensional matrix of relatively similar length and width g. In line 11, the file is converted into an array of unsigned integers, and in lines 13–14, the file is deleted from the directory after the array has been reshaped and the file has been closed so that, once the function is executed, it does not preprocess the same module repeatedly. In lines 16–20, the values of the three channels are obtained and sequentially output as an RGB image. In line 17, each value of the 2D matrix is used as the red channel value, in line 18, 60 times the local entropy value of each value of the 2D matrix is used as the green channel value, the sliding window is set to 3*3 size, in line 19, 60 times the global entropy value of each value of the 2D matrix is used

as the blue channel value, in line 20, the RGB image object is created, and in line 21, the RGB image is Save the RGB image to the specified folder. In lines 22–26 reshape the image obtained above to a specified size of 100*100 and save it to a folder, in line 22 convert the data type of the variable image to an 8-bit unsigned integer type, in line 23 reshape the variable image to an image of size 100*100 by using the double interpolation algorithm, store all the obtained images of the same size h to the Wasm_images folder and lines 25–26 detect all the images in Wasm_images folder. Once the binary file is converted to an image and added to this folder, the detector() function is called which takes the image as an argument. This function will be referenced and discussed in the following subsections the preprocess function ends with a recursive call, ensuring that it continues to check the directory for new input every second (Table 1).

3.3 Training and Detection of Image Classification Models

The detector() function will call the above processed image as input to the CNN model. The model is pre-trained on a dataset consisting of 55 malicious WASMs and 424 benign WASMs after binary data visualization. A total of three convolutional layers, three pooling layers and their two fully connected layers are used in this method to form the trained CNN model and the structure is shown in Fig. 3. The convolutional layer is the core of the convolutional neural network, which is mainly used to extract the surface and deep features of the image. The number and size of convolution kernels of the first convolutional layer used in the present invention are 20,3*3, the second layer is 50,3*3, and the third layer is 100,3*3. The pooling layer is mainly used for feature selection and downsampling of the image. The pooling layer used in the present invention has a size of 2*2 and a step size of 2. The features are fed into the fully connected layer after feature extraction and feature selection through the convolutional layer and the pooling layer. The role of the fully connected layer is mainly to reduce the influence of the spatial location of the image on the features and to classify the samples. In this invention, two fully connected layers are used, 1*256,1*2 respectively, and finally softmax is used to output the classification of the image. In addition, in this method, the output of each layer is nonlinearly transformed by Relu function.

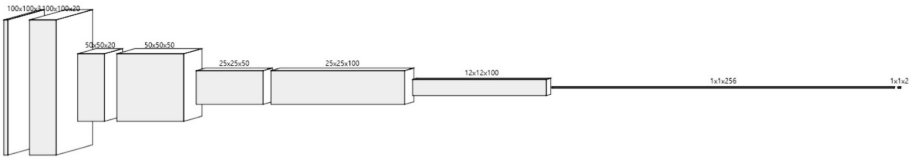


Fig. 3. CNN model architecture diagram

The trained model is used to detect random web pages in the real world. Access the web page url thus extracting the WASM binary file from the web page,

Table 1. Wasm module visualization methodology

Algorithm 1: Wasm module visualization methodology

```
def preprocess():
    while len(Wasm directory) == 0 do
        time.sleep(1)
    if len(Wasm directory)! = 0 then
        Wasm__images → []
        for file in Wasm directory do
            f → open(file)
            ln → getSize(file)
            width → math.pow(ln, 0.5)
            rem → ln%width
            a → array(B)
            a.fromfile(f, ln  rem)
            f.close()
            os.remove(file)
            g → reshape(len(a)/width), width)
            rgb__image→np.zeros(len(a)/width, width, 3)
            rgb__image[:,:,0]→ g
            rgb__image[i, j, 1] → int(local__entropy * 60)
            rgb__image[:,:,2] → int(global__entropy * 60)
            image → Image.fromarray(rgb__image)
            image.save(output__path)
            image→ np.uint8(image)
            h → Image.fromarray(image).resize((100, 100)
            h.save(Wasm__images)
    detector(Wasm images)
return preprocess()
```

convert the WASM file into an RGB image as per the visualization in the previous section and feed it into the trained model, if the mining web page detection model classifies the WASM binary file as malicious then the user is informed that the web page they are currently accessing is mining cryptocurrencies using their computational resources and they are advised to close the webpage, thereby terminating any mining processes running in the background. However, if the binary is classified as benign, the detection module performs no action, the user continues to browse the page, and the collection module continues to check the instantiation of the WASM module in the web page.

4 Performance Evaluation

To evaluate the effectiveness of the NoCrypto method, we conducted experiments on two different datasets, which include WASM binary files and web pages used for mining attack detection. The first dataset consists of WASM binary files used to train the NoCrypto model. The second dataset comprises real-world internet web pages that utilize WASM. Below, we provide a detailed description of each dataset, followed by an explanation of the metrics used in our evaluation:

Training dataset: the dataset used to train the WASM detector consists of 55 malicious and 424 benign WASM binaries obtained from a large number of studies and resources. A large portion of the dataset consists of binaries collected and used by studies in the literature: MinerRay [13], Minesweeper [4]. The remaining binaries were collected manually using resources such as the MadeWithWasm [14] website, the webassembly-examples-main [15] project, etc. A breakdown of the number of binaries collected from each resource, including the distribution of benign and malicious samples, can be found in Table 2 below. This is followed by a brief description of the other resources and how binaries were collected from each.

Table 2. Description of the training data set

	Benign WASM	Malware WASM
MinerRay	141	21
Minesweeper	4	34
Made with Wasm	263	0
Webassembly-examples-main	16	0

MadeWithWASM: Made With WebAssembly is a site that showcases applications, projects, and websites that use WebAssembly. By accessing the projects on the site, access is made using Chrome DevTools, which downloads the WASM module in WAT text format and converts it to a binary file using WABT. Note that when checked with Chrome DevTools, not all use cases and websites listed on MadeWithWASM instantiated the WASM module. At this point download all the WASM modules by accessing all the files for the project from the github site and add them to the dataset of benign WASM samples.

Webassembly-examples-main: This is a github project containing web files that use WASM.

Test dataset: this dataset consists of real websites, both benign and encrypted, which use WebAssembly technology. The source of this dataset is Tranco [16].

Tranco is a public top-level domain ranking dataset based on Alexa rankings for measuring and analyzing the most popular websites on the Internet. It provides a list of millions of domains, ranked by their number of visits and

traffic. We extracted 123 wasm binaries from the top 100,000 ranked pages on the Tranco website. 119 of them are benign,4 are malicious.

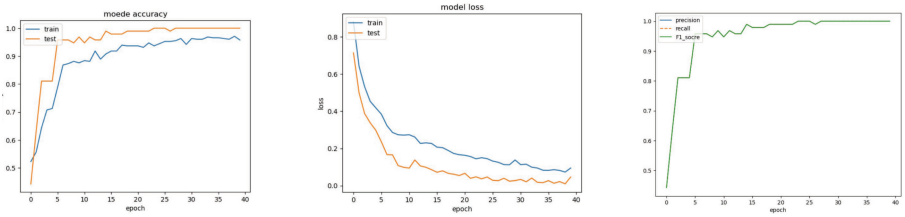
In order to label the real dataset we have used certain criteria to label the real samples and the criteria used are as follows:

(1) The obtained wasm binary file is converted to WAT text format by using the WABT tool to check whether the hash function exists in the WAT file.
(2) Since the size of the WASM binary of the game/application is significantly larger than the WASM binary of the malicious application, the size of the extracted WASM binary was tagged by checking the size of the website.

Classification Accuracy Metrics: To evaluate the accuracy of NoCrypto, we selected a number of accuracy metrics. Using the training dataset, we evaluated the performance of NoCrypto by calculating its overall accuracy, learning curve (loss vs. number of calendar elements), and ROC curve (true positive rate vs. false positive rate). The NoCrypto framework is further evaluated by testing the field dataset using metrics such as accuracy, precision, recall and F1 score.

4.1 Performance of NoCrypto Detection Methods on Training Set

NoCrypto's performance on the training dataset is evaluated based on a number of metrics, including accuracy, optimization loss, true positive rate, and false positive rate. The dataset is divided into a training set and a test set using an 80/20 split. In Fig. 4(a), it can be seen that the model converges to 100% accuracy on the test set. The optimization learning curve in Fig. 4(b) shows that after about 30 epochs, both the training and testing losses drop to a stable point. This suggests that the model is neither over nor underfitting and therefore can be effectively generalized. Figure 4(c) shows the model performance of the detector. The F1 Score, Recall and Precision are all equal and their values are converging to 100%, which means that the model performs very well when combining Precision and Recall, effectively identifying positive examples and excluding negative ones with very few false predictions.



(a) Accuracy (b) Learning Curve (c) ROC Curve

Fig. 4. Model Training Performance Metrics

4.2 Performance of NoCrypto Detection Methods On a Test Set

The performance evaluation metrics on the test set are the same as the training set, by scanning the top 100,000 websites ranked by Tranco website, 389 WASM binaries are obtained, some web pages use the same WASM module, a total of 123 WASM binaries are obtained after de-emphasis, a total of 119 WASM binaries are generated by manually executing each web page tagging the 123 files, 119 benign web pages, 4 malicious web pages, and 10 tests finally obtain an average of 99.18% accuracy, 1 benign sample is incorrectly identified as malicious, and 4 malicious web pages are generated as WASM binary files and 4 WASM binary files generated by malicious web pages, 10 tests were performed finally obtaining an average accuracy of 99.18%, 1 benign sample was incorrectly identified as malicious and all 4 malicious web pages were correctly identified. Table 3 shows other calculated accuracy metrics that illustrate the performance of the NoCrypto system for this dataset.

Table 3. Performance of the NoCrypto system on the test set

Evaluation indicators	Accuracy	Precision	Recall	F1 Score
NoCrypto	99.18%	99.18%	99.18%	99.18%
MINOS	96.88%	96.88%	96.88%	96.88%

In order to verify the classification performance of the method in this paper, we compare it with the latest visualization-based mining webpage detection method that also uses the above dataset. The accuracy of the above dataset tested by the grayscaling method mentioned in this MINOS [12] paper reaches only 96.88%, which also shows that the visualization method in our detection method is more effective and the detection model is more robust.

4.3 Performance of the NoCrypto Assay in Confounded Samples

To evaluate the robustness of NoCrypto against adversarial evasion, we collected WASM mining obfuscation samples from publicly available papers that employ function name obfuscation, which obfuscates a varying number of function names in source and header files. Figure 5 shows a grayscale image representation of the mining samples. As shown in the figure, the resultant binaries of mining have very similar grayscale image representations, despite blurring different numbers of function names in the mining code. To evaluate the robustness of NoCrypto to the blurred mining samples shown in Fig. 5, we sent the samples to NoCrypto and observed that NoCrypto was able to detect all blurred binaries regardless of the level or degree of blurring.

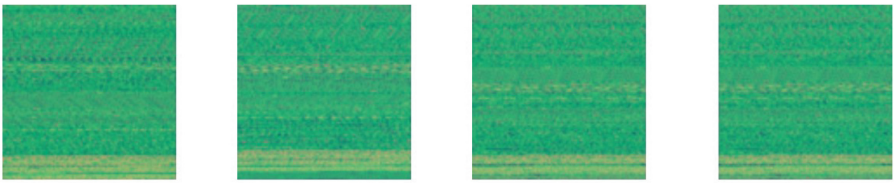


Fig. 5. RGB image after visualization of WASM mining samples with increasing degree of confusion

5 Conclusions

Due to the increasing prevalence of mining web pages utilizing new WebAssembly (WASM) technology, traditional static detection methods are prone to interference from obfuscation techniques, while dynamic detection techniques require significant computational resources. In this paper, we propose a novel static detection method that visualizes WASM binary files as RGB images and utilizes a CNN network model for detection and recognition. This method achieves excellent results, demonstrating a certain degree of resistance to obfuscation techniques, fast execution speed, and minimal resource consumption during operation. In the future, we plan to incorporate a detection method for mining web pages based on JavaScript (JS) scripts. Ultimately, we aim to develop a browser extension for Google Chrome that detects mining websites. This extension will provide users with a comprehensive solution to identify and protect against the unauthorized utilization of their computing resources for mining activities. The extension will be designed for widespread adoption and usage.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under Grant No. 61862022, 62172182; the Natural Science Foundation of Guangdong under Grant No. 2023A1515011084; the Key Scientific Research Platforms and Projects of Guangdong under Grant No. 2022ZDZX1022; the Talent fund of Guangdong Polytechnic Normal University under Grant No. 99166990223; the Doctoral Program Establishment and Construction of Scientific Research Capacity Improvement Project of Guangdong Polytechnic Normal University under Grant No. 22GPNUZDJS27; the fund of Key Laboratory of Information Security of Zhejiang Province under Grant No. KF202306.

References

1. Lehmann, D., Pradel, M.: Wasabi: a framework for dynamically analyzing webassembly. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 1045–1058. Association for Computing Machinery, NY, United States (2019)
2. Musch, M., Wressnegger, C., Johns, M., Rieck, K.: New kid on the web: a study on the prevalence of WebAssembly in the wild. In: Perdisci, R., Maurice, C., Giacinto, G., Almgren, M. (eds.) DIMVA 2019. LNCS, vol. 11543, pp. 23–42. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22038-9_2

3. Tekiner, E., Acar, A., Uluagac, A.S., Kirda, E., Selcuk, A.A.: SoK: cryptojacking malware. In: 2021 IEEE European Symposium on Security and Privacy, pp. 120–139. IEEE, Piscataway, NJ (2021)
4. Konoth, R.K., 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, pp. 1714–1730. Association for Computing Machinery, NY, United States (2018)
5. Wang, W., Ferrell, B., Xu, X., Hamlen, K.W., Hao, S.: SEISMIC: secure in-lined script monitors for interrupting cryptojacks. In: Lopez, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018. LNCS, vol. 11099, pp. 122–142. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98989-1_7
6. Bian, W., Meng, W., Zhang, M.: Minethrottle: defending against wasm in-browser cryptojacking. In: Proceedings of the Web Conference 2020, pp. 3112–3118. Association for Computing Machinery, NY, United States (2020)
7. Yu, G., Yang, G., Li, T., Han, X., Guan, S., Zhang, J., Gu, G.: MinerGate: a novel generic and accurate defense solution against web based cryptocurrency mining attacks. In: Lu, W., et al. (eds.) CNCERT 2020. CCIS, vol. 1299, pp. 50–70. Springer, Singapore (2020). https://doi.org/10.1007/978-981-33-4922-3_5
8. Bian, W., Meng, W., Wang, Y.: Poster: detecting webassembly-based cryptocurrency mining. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2685–2687. Association for Computing Machinery, NY, United States (2019)
9. Rüth, J., Zimmermann, T., Wolsing, K., Hohlfeld, O.: Digging into browser-based crypto mining. In: Proceedings of the Internet Measurement Conference 2018, pp. 70–76. Association for Computing Machinery, NY, United States (2018)
10. Kharraz, A., et al.: Outguard: detecting in-browser covert cryptocurrency mining in the wild. In: The World Wide Web Conference, pp. 840–852. Association for Computing Machinery, NY, United States (2019)
11. Tommasi, F., Catalano, C., Corvaglia, U., Taurino, I.: MinerAlert: an hybrid approach for web mining detection. J. Comput. Virol. Hacking Tech. **18**(4), 333–346 (2022)
12. Naseem, F.N., Aris, A., Babun, L., Tekiner, E., Uluagac, A.S.: MINOS: a lightweight real-time cryptojacking detection system. In: NDSS. At: Virtual (2021). https://doi.org/10.14722/ndss.2021.24444
13. Romano, A., Zheng, Y., Wang, W.: Minerray: semantics-aware analysis for ever-evolving cryptojacking detection. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pp. 1129–1140. Association for Computing Machinery, NY, United States (2020)
14. Made With Webassembly. https://madewithwebassembly.com/. Accessed 21 Mar 2023
15. Webassembly-examples-main. https://github.com/mdn/webassembly-examples/. Accessed 01 Apr 2023
16. Pochat, V.L., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., Joosen, W.: Tranco: a research-oriented top sites ranking hardened against manipulation. In: arXiv preprint arXiv:1806.01156 (2018)