

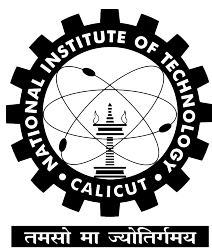
# MinerJAB: A Tool for detection and mitigation of Host-based Cryptojacking Malware

CS4099D Project Final Report

*Submitted by*

|                    |           |
|--------------------|-----------|
| Bharath Sajan      | B200016CS |
| Jackson Stephan    | B200743CS |
| Mohamed Afthab E K | B200719CS |

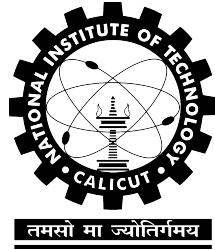
Under the Guidance of  
Dr.Vasudevan A.R.



Department of Computer Science and Engineering  
National Institute of Technology Calicut  
Calicut, Kerala, India - 673 601  
May 07, 2024

**NATIONAL INSTITUTE OF TECHNOLOGY  
CALICUT, KERALA, INDIA - 673 601**

**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**



2024

## **CERTIFICATE**

*Certified that this is a bonafide record of the project work titled*

**MINERJAB:TOOL FOR DETECTION AND MITIGATION OF  
HOST-BASED CRYPTOJACKING MALWARE**

*done by*

**Bharath Sajan**

**Jackson Stephan**

**Mohamed Afthab E K**

*of eighth semester B. Tech in partial fulfillment of the requirements for the  
award of the degree of Bachelor of Technology in Computer Science and  
Engineering of the National Institute of Technology Calicut*

**Project Guide**


**Dr. Vasudevan A R**

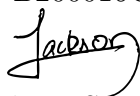
**(Assistant Professor)**

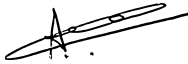
# DECLARATION

We hereby declare that the project titled, **MinerJAB:Tool for Detection and mitigation of Host-based Cryptojacking malware**, is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or any other institute of higher learning, except where due acknowledgement and reference has been made in the text.

Place : NIT Calicut  
Date : 07-05-2024

Signature:   
Name : Bharath Sajan  
Reg. No. : B200016CS

Signature:   
Name : Jackson Stephan  
Reg. No. : B200743CS

Signature:   
Name : Mohamed Afthab E K  
Reg. No. : B200719CS

## **Abstract**

The rise in demand for cryptocurrencies built on blockchain technology has led to a parallel spike in illegal mining operations that produces digital cash. Because cryptocurrency mining requires a lot of resources, bad actors use a variety of strategies, such as secret techniques and unauthorized downloads, to stealthily mine cryptocurrency on victims' devices without their awareness. As part of our research, we offer MinerJAB, a novel method for detecting host-based cryptojacking malware that blends behavior-based dynamic analysis with hash based detection utilizing five key cryptomining features. In this study, we investigate three distinct approaches in detail: a sequential analysis approach, random forest classification, and neural networks. We meticulously examine these methodologies using the same malware dataset and features to ensure a comprehensive comparison.

## **ACKNOWLEDGEMENT**

We would like to express our sincere and heartfelt gratitude to our guide and mentor Dr. Vasudevan A R, who has guided us throughout the course of the final year project. Without his active guidance, help, cooperation and encouragement, we would not have made headway in the project. We would like to thank our parents and the faculty members for motivating us and being supportive throughout our work. We also take this opportunity to thank our friends who have cooperated with us throughout the course of the project.

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                             | <b>2</b>  |
| <b>2</b> | <b>Literature Survey</b>                        | <b>5</b>  |
| 2.1      | Static Methods . . . . .                        | 5         |
| 2.2      | Dynamic Methods . . . . .                       | 5         |
| 2.3      | ML-based Methods . . . . .                      | 7         |
| <b>3</b> | <b>Problem Definition</b>                       | <b>10</b> |
| <b>4</b> | <b>Methodology</b>                              | <b>11</b> |
| 4.1      | Design Overview . . . . .                       | 11        |
| 4.2      | Hash-based Detection . . . . .                  | 13        |
| 4.3      | Feature Identification and Extraction . . . . . | 13        |
| 4.3.1    | Feature Identification: . . . . .               | 13        |
| 4.3.2    | Feature Extraction : . . . . .                  | 18        |
| 4.3.3    | Data Collection and Sandboxing: . . . . .       | 20        |
| 4.4      | Dynamic Analysis: . . . . .                     | 21        |
| <b>5</b> | <b>Results and Comparisons</b>                  | <b>28</b> |
| <b>6</b> | <b>Conclusions and Future work</b>              | <b>30</b> |
|          | <b>References</b>                               | <b>31</b> |

# List of Figures

- 1.1 A Visual Representation of Host-Based Cryptojacking[8] . . . 3
- 4.1 High level diagram of the MinerJAB . . . . . 12
- 4.2 Variation of CPU usage of a host-based cryptojacker with time 19
- 4.3 Dynamic Analysis Overview . . . . . 22
- 4.4 Sequential Analysis Overview . . . . . 24
- 4.5 Relative importance of features with respect to Random Forest  
Classification . . . . . 25

# List of Tables

|     |   |    |
|-----|---|----|
| 2.1 | Summary of Reviewed Detection Methods . . . . .   | 9  |
| 4.1 | List of Cryptographic libraries called by Cryptojackers[7] . . .                                | 17 |
| 4.2 | Classification Report from Random Forest Classification . . .                                   | 26 |
| 4.3 | Confusion Matrix of the Random Forest Classification . . . . .                                  | 26 |
| 4.4 | Results from the Deep Learning Model . . . . .  | 27 |
| 4.5 | Confusion Matrix of the Deep Learning Model . . . . .   | 27 |
| 5.1 | Comparison of results of previous works on cryptojacking de-<br>tection with MinerJAB . . . . . | 28 |



# Chapter 1

## Introduction

In recent years, the world has witnessed a remarkable surge in the popularity of cryptocurrencies, driven by the advent of Web 3.0 and the revolutionary blockchain technology. With the high demand and increasing value of cryptocurrencies, the practice of mining has gained significant traction. Cryptocurrency mining is the process of validating and adding transactions to a blockchain by solving complex cryptographic puzzles, and miners are rewarded with cryptocurrency tokens for their efforts. This surge in cryptocurrency mining has also given rise to a phenomenon known as **cryptojacking**.

Cryptojacking can be roughly defined as the unauthorized usage of a victim's computational power, which results in profits for the attacker, commonly in the form of cryptocurrencies such as Bitcoin. This malicious activity first came to public attention in 2017, with the emergence of Coinhive. What could have been a potentially harmless and profitable way for website owners to monetize their sites or provide services became a means for attackers to mine cryptocurrency without the user's consent. This was achieved by injecting code into legitimate websites and exploiting the computational resources of unsuspecting visitors. In one major instance, cryptojacking malware was merged with Google's advertising packages on YouTube, leading to significant concerns and heightened awareness in the digital security landscape[1].

Cryptojacking attacks can generally be classified into two main types:

**Host-Based Cryptojacking:** This type of attack involves the unauthorized usage of a victim's computing resources, often involving the injection of malicious code into a host system. Deep learning approaches have been employed for host-based cryptojacking malware detection, enhancing the capability to identify and mitigate such threats[2]. Figure 1. depicts the lifecycle of host-based cryptojackers[8].

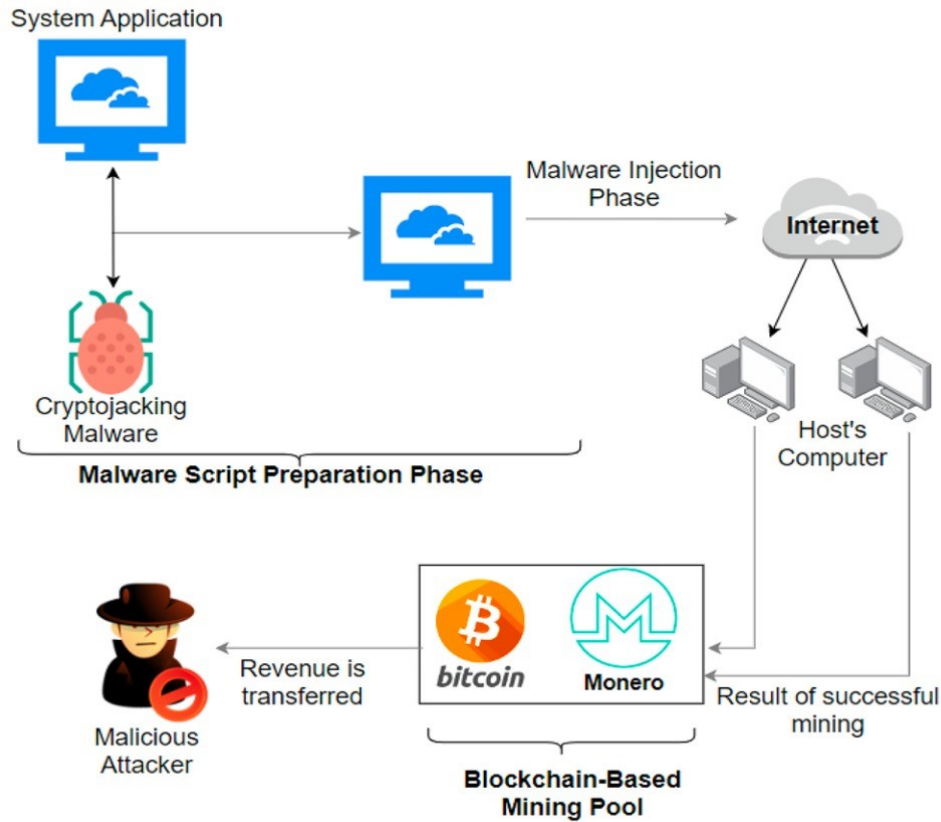


Figure 1.1: A Visual Representation of Host-Based Cryptojacking[8]

**In-Browser Cryptojacking:** In-browser cryptojacking occurs when the CPU power of a website's visitors is hijacked to perform CPU-intensive cryp-

tocurrency mining. However, due to the discontinuation of services like Coinhive, in-browser cryptojacking has become less prevalent. As a result, our focus will primarily be on host-based cryptojacking attacks[3].

Although cryptojacking is a relatively new form of malware, its impact and prevalence have grown significantly. There has been substantial research and analysis of cryptojacking attacks, their detection, mitigation, and preventive measures. For further insights into this field, the aim here is to conduct a comprehensive study and contribute to the knowledge and tools available for the detection and mitigation of host-based cryptojacking attacks.

# Chapter 2

## Literature Survey

This section reviews existing detection approaches for cryptojackers, which can be broadly categorise thme into 3 main categories.

### 2.1 Static Methods

One of the first and most widespread detection approaches for malware in general is the analysis of static signatures. Several tools, such such as Dr.Mine[4] and MinerBlock[5] implement static methods to detect mining activities and blacklist malicious web sites. The advantage of static analysis is a high degree of reliability upon detection. The disadvantage is that it's useless if there's no signature within the database, making it unusable against zero-day attacks.

### 2.2 Dynamic Methods

Another prominent method is Dynamic analysis, which studies behavior of an already running program. It tries to determine whether the behavior of the program is malicious or not and is thus also called behavioral analysis.

Behavior-Based Detection of Cryptojacking Malware[6] discussed a behaviour-based detection model monitoring CPU load, RAM and average quadratic deviation of CPU utilization and was tested on a virtual machine with Windows 7, 3GHz single-core CPU and 4GB RAM. To obtain data about running processes and applications the authors suggested using Windows Management Instrumentation(WMI), as it operates on the kernel level, it can avoid some stealth techniques employed by cryptojackers. The model showed 81% accuracy for over 50 samples including both browser-based and file-based cryptojacking malware. It achieved only limited success and wasn't able to detect cryptojacking malware with dynamic CPU load or advanced stealth techniques.

The same authors developed a more advanced-behaviour model in [7] considering enhancements such as multi-core CPU usage, network access, cryptographic library calls and lowering initial requirement of CPU usage from 30% to 10%. The research states that the key difference between a legitimate network connection and a cryptojacking malware connection is the need to access the network on a constant basis with more than 50% samples attempted to access network every 5 seconds or less. The program achieved an accuracy of 93% against 100 more cryptojacking malware samples on a VirtualBox virtual machine with following characteristics: Windows 10 OS, quad-core 3GHz CPU and 8GB RAM. The program didn't differentiate between legitimate and malicious miners for obvious reasons and didn't focus on detection of GPU-based malicious miners.

Dynamic approaches have been proven more effective and efficient than static ones affected by obfuscation techniques. Furthermore, static approaches are ineffective against unseen cryptojackers.

## 2.3 ML-based Methods

Besides these, machine learning techniques have gained popularity recently. The model in [12] uses a convolutional neural network (CNN) to extract features from the system call traces. A CNN is a type of artificial neural network that can learn from data by applying filters and pooling operations to create feature maps. The CNN can learn to recognize important features from the system call traces, such as the frequency, duration, and sequence of system calls, and how they differ from normal processes. The model then uses a recurrent neural network (RNN) to classify the processes as benign or malicious. An RNN is a type of artificial neural network that can process sequential data by maintaining a hidden state that stores information from previous inputs. The RNN can learn to capture the temporal dependencies and context of the system call traces and use them to make accurate predictions about the nature of the processes.

The model has been evaluated on a dataset of 129,380 samples, which consist of 64,690 benign samples and 64,690 malicious samples. The malicious samples include different types of cryptojacking malware, such as Coinhive, Crypto-Loot, JSEcoin, Webminepool, and XMRig. It has achieved an overall accuracy of 98%, which means that it can correctly classify 98% of the samples as benign or malicious and also has a low performance overhead, which means that it does not consume much resources or affect the normal functioning of the host machine while running in the background. It is highly scalable, which means that it can handle large amounts of data and adapt to new types of cryptojacking malware that may emerge in the future. The model is also robust, which means that it can resist evasion techniques and noise that may be used by cryptojackers to avoid detection.

Paper [10] introduces CJDetector, a host-based cryptojacking detection system designed to identify malicious processes covertly mining cryptocurrencies on a user's computer. The system monitors CPU usage across all processes, filtering out those exhibiting sustained high CPU utilization. It then

scrutinizes the function call information of these filtered processes, employing machine learning algorithms to categorize them as either cryptojacking or benign based on their system call patterns. CJDetector promptly alerts users upon detecting any cryptojacking activity and facilitates the termination of the offending process.

Another novel approach to detect both host-based and in-browser cryptojacking was proposed by Ganapathy Mani et al[13]. In this approach, assembly code executed at specified intervals of time is analyzed using deep learning techniques. It then filters out instructions involved in cryptomining operations, such as bitwise operations (XOR, AND, OR), encryption (AESENC), and cryptographic computations (AESKEYGENASSIST). The authors implemented this approach, and the overall mean time to detect (MTTD) was observed to be less than 10 milliseconds.

The Naïve Bayes Cryptojacking Detection model, as defined in paper [11], utilizes threshold values to classify software as benign or malicious. It takes CPU utilization, RAM usage, network transfer rate, and the number of cryptographic API calls as inputs. Conditional probabilities are defined for these features, allowing the model to make classifications. It calculates the aggregated conditional probability (\*k) for both benign and malicious classes using Bayes' theorem. Subsequently, it classifies the software based on the higher probability

Lastly the paper [9] which proposes a new cryptomalware detection mechanism based on monitoring the CPU usage of visited web pages. The authors argue that this is a more reliable way to detect mining malware than simply monitoring overall CPU usage, as many web applications have high CPU usage for legitimate reasons. Additionally, the authors note that mining malware often throttles CPU usage to stay below the radar, so a simplistic mechanism that simply triggers an alarm if CPU usage goes above a certain threshold is likely to produce false positives. The proposed mechanism works by collecting a set of CPU metrics from the visited web pages, such

as CPU usage, memory usage, and network traffic. These metrics are then fed into a machine learning algorithm to classify the page as either malicious or legitimate. The authors evaluated the proposed mechanism on a dataset of over 100,000 web pages, and achieved a precision and recall of over 99%. This means that the mechanism was able to correctly identify malicious web pages with very few false positives or negatives. Overall, the paper presents a promising new approach to cryptojacking detection. The proposed mechanism is able to achieve high accuracy even in the presence of malware that throttles CPU usage. The authors also provide a detailed evaluation of their mechanism, which is helpful for assessing its performance in real-world conditions.

Table 2.1: Summary of Reviewed Detection Methods

| Detection Technique          | Type       | Utilized Data                              | Notes   |
|------------------------------|------------|--|---|
| Static Analysis [4], [5]     | Both       | Signatures                                 | Not effective against obfuscations                                  |
| Behavioral Analysis [6], [7] | Host-Based | CPU load, RAM, network access              | Limited success against sophisticated techniques                    |
| CNN and RNN[9]               | Host-Based | System call traces                         | Robust against evasion techniques and noise                         |
| Deep Learning[11]            | Both       | Assembly code                              | Achieves fast Mean Time to Detect (MTTD)                            |
| Machine Learning[10]         | Web-Based  | CPU metrics, memory usage, network traffic | Maintains high precision and recall, even with CPU usage throttling |



# Chapter 3

## Problem Definition

Our objective is to create a complete solution for identifying and preventing host-based cryptojacking malware through the use of hash-based detection and behavior-based techniques. Furthermore, we aim to do a thorough comparative study of 3 different methodologies using identical datasets and features to determine the most effective methodology.

# Chapter 4

## Methodology

### 4.1 Design Overview

The MinerJAB is a host-based cryptojacking malware detection and mitigation tool, fundamentally comprising of 2 concurrently running program with the respective functions: Hash-based detection and Behaviour based detection.

The hash-based detection process validates the identity of newly downloaded files by cross-referencing them against a database of known cryptojacking malware signatures, utilizing SHA-256 hashes. However, this method may falter in detecting newly crafted or obfuscated malware variants. To address this limitation, the dynamic analysis process comes into play. If a downloaded file, upon execution, exhibits CPU usage exceeding 10%, it triggers the dynamic analysis model. This model runs for a defined period, scrutinizing the file's behavior to assess its potential threat level.

Unlike other types of malware, cryptojacking malware poses less immediate danger and can operate on the system for a short duration without any consequences. Upon successful detection, the process is terminated, and our

system continues to concurrently monitor for any newly added files or CPU spikes. By employing hash-based detection, we ensure that our dynamic analysis remains lightweight, thus enhancing the efficiency of the MinerJAB.

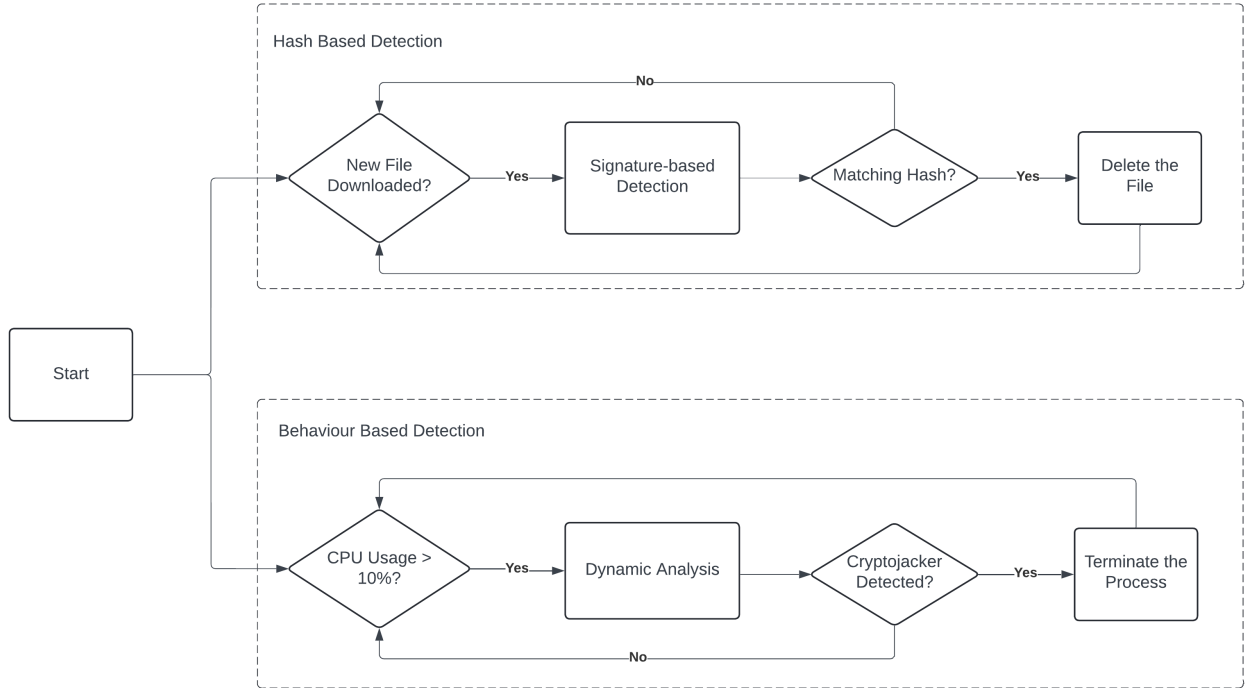


Figure 4.1: High level diagram of the MinerJAB

To comprehensively elucidate the model's operation, we segment this research into 3 phases:

Phase 1: Hash-Based Detection

Phase 2: Feature Identification and Extraction for Dynamic Analysis

Phase 3: Dynamic Analysis

In the dynamic analysis phase, the 3 following methodologies are used:

- Sequential Model
- Random Forest Classifier

- Neural Network Classifier

## 4.2 Hash-based Detection

During this initial phase, the system meticulously examines newly arrived files within the "Downloads" directory. Leveraging a comprehensive database comprising 74,000 signature samples of malware, the SHA256 hash of each executable file is cross-referenced against known cryptojacking malware signatures obtained from sources such as VirusTotal, VirusShare, and Malware-Bazaar. If the SHA256 hash of the newly added file matches an entry in the database, a message is displayed in the terminal, and the file is promptly removed from the directory. SHA256 hash is widely favored in malware identification due to its ubiquity in file authenticity checks.

This hash-based detection process proficiently identifies instances of malware without necessitating execution, thereby significantly alleviating the workload for our dynamic model. Files whose hash values do not correspond to any existing entries in the database proceed to undergo dynamic analysis.

## 4.3 Feature Identification and Extraction

### 4.3.1 Feature Identification:

The 5 most crucial characteristics for identifying cryptojacking malware have been integrated into all models following an extensive study. The insights derived from Advanced Behavior [2] greatly influenced this decision.

◇ CPU Usage:

The CPU is essential to the cryptocurrency mining process because it is in charge of carrying out the computing activities required. Since cryptomining requires high CPU utilization, the computational intensity of the mining process is essential.

Cryptojackers use a variety of strategies to mask their CPU utilization in an effort to evade discovery. Depending on system use or the time of day, one popular strategy is to dynamically modify the intensity of cryptomining operations. Cryptojackers try to reduce suspicion and avoid detection by using less CPU power during peak hours or during periods of high system activity.

Another tactic is to use mining algorithms that are stealthy and use less processing power to provide revenue for the attacker. These algorithms can make it more difficult for detection techniques to identify suspicious activity based solely on CPU consumption by giving priority to low-impact mining operations or using alternate consensus procedures that use less CPU power.

CPU utilization for a process is typically calculated by the operating system's scheduler. It's a measure of the amount of time a process is running (i.e., using the CPU) compared to the total available CPU time.

Mathematically, it can be calculated as follows:

$$\text{CPU Utilization} = \left( \frac{\text{CPU Time used by the process}}{\text{Total CPU Time available}} \right) \times 100\%$$

Here:

CPU Time used by the process is the amount of time the CPU was busy executing instructions of the process. This is also known as CPU time or processor time. It does not include time spent waiting for resources (like I/O operations), even if the process is running.

Total CPU Time available is the total amount of time the CPU has been running. This includes time spent executing all processes and time spent idle.

This calculation gives a percentage that represents how much of the CPU's capacity was used by the process.

◇ Network Transfer Rate:

Cryptojacking operations require a functional network connection in order to interface with mining pools and transfer data. Security experts can spot unusual activity suggestive of cryptojacking activities by examining upload and download traffic patterns. Studies suggest specific thresholds, including an upload data threshold of 6KB/min and an average of 300KB/min, to identify suspicious activities. Research indicates that in order for cryptomining operations to be successful, a network connection must occur every 40 seconds.

◇ RAM Usage:

Cryptomining software requires certain amount of libraries and DLLs to be loaded into the RAM to solve to carry out mining operations.

For several reasons, cryptominers often utilize high RAM:

- **Enhanced Mining Efficiency:**

- To solve challenging cryptographic puzzles and verify blockchain transactions, cryptomining software needs a lot of processing power.
- Cryptojackers can enhance the efficiency of their mining activities and outperform other miners by employing large RAM capacities. This allows them to process transactions faster.

- **DLL Injection Techniques:**

- To run its malicious code inside the address space of other programs that are legitimately operating on the victim's machine, cryptomining malware frequently makes use of DLL injection techniques.
- This entails inserting the cryptomining logic into a different DLL file and putting it into a trusted process's memory, such a system service or web browser.

- The injected DLL continues to reside in memory, hence increasing RAM consumption overall.

- **Optimization of RAM Usage:**

- The presence of DLLs in memory enables cryptomining malware to optimize RAM usage for mining operations.
- DLLs may implement mining algorithms, handle network communication with mining pools, and manage shared memory buffers for storing mining-related data.
- By leveraging DLLs, cryptomining malware can modularize its functionality and efficiently utilize RAM resources for increased mining efficiency.

RAM facilitates the temporary execution of mining processes, which is central to the operation of cryptomining malware. Once a system is compromised, the malware typically loads into memory and commences mining algorithms using the CPU and GPU resources of the compromised system. Additionally, the malware may create temporary RAM buffers and data structures to expedite the mining process and store intermediate results.

- ◇ Average CPU quadratic deviation:

Average quadratic deviation of CPU usage is crucial because cryptojacking malware frequently modifies its CPU usage to evade detection. These variations in CPU utilization can be minor but provide important clues to cryptojacking activity, which emphasizes the necessity for advanced detection methods.

The average quadratic deviation of CPU utilization can be analyzed to provide insights into possible fluctuations, which are frequently indicative of cryptojacking activity. By measuring the dispersion or variability of CPU utilization over time, this statistical indicator enables academics and

security experts to spot unusual patterns that could be signs of cryptomining activity. By making the distinction between typical fluctuations in CPU utilization and unusual conduct linked to cryptojacking, the average quadratic deviation facilitates more precise identification and reduction of malicious activities. Thus, it serves as a robust statistical tool in studies aimed at understanding and combating cryptojacking threats.

◇ Cryptographic Libraries:

Cryptomining fundamentally relies on cryptographic functions and libraries for mining operations which involve, hashing, encryption and decryption. Research has identified common cryptographic functions and system calls associated with cryptojacking activities.

Table 4.1: List of Cryptographic libraries called by Cryptojackers[7]

| Library              | Description   |
|----------------------|---|
| bcryptprimitives.dll | Windows cryptographic primitives library            |
| crypt32.dll          | Crypto API32  |
| cryptbase.dll        | Base cryptographic API DLL                          |
| cryptdll.dll         | Cryptography manager                                |
| cngaudit.dll         | Windows cryptographic next generation audit library |
| rsaenh.dll           | Microsoft enhanced cryptographic provider           |
| cryptsp.dll          | Cryptographic Service Provider API                  |

Monitoring loaded libraries and detecting the presence of these 7 specific functions provides valuable evidence of potential malicious activity.



### 4.3.2 Feature Extraction :

All the features are observed for a time-frame of 40 seconds, which are then written into a CSV file.

#### 1. CPU Usage and Quadratic Deviation:

To extract CPU usage details, the **psutil** library in python is used to query the system for real-time information on CPU utilization.

Using CPU usage, average CPU quadratic deviation is calculated using the formula given below.

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (CPU_i - \overline{CPU})^2}$$

This metric provides valuable insights into the dynamic behavior of the system's CPU resources and enables us to make informed decisions regarding system management and optimization strategies.

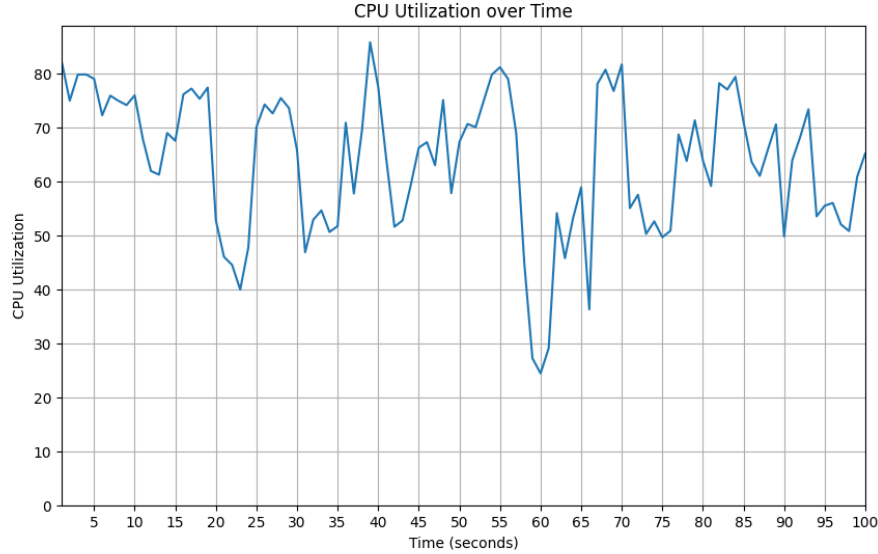


Figure 4.2: Variation of CPU usage of a host-based cryptojacker with time

## 2. Network Transfer Rate :

The **Scapy** library is utilized for packet sniffing and real-time analysis of network traffic. Its effectiveness in capturing and analyzing network packets, particularly on Windows systems, is augmented by its dependency, **Npcap**. In this context, Scapy is employed to capture network packets and discern uploads based on their source MAC address. The metric "Network Transfer Rate" is computed as the sum of the upload speed and download speed, measured in Bytes/sec.

## 3. Ram Consumption :

Ram usage is also calculated using **psutil** library in Megabytes. When assessing RAM consumption, psutil employs process identifiers (PIDs) to enumerate all active processes, akin to an inventory of tasks being executed by the CPU. This metric excludes the memory that has been

swapped out to disk, thus providing a precise measure of the active memory footprint of a process.

4. Cryptographic Library Call Counter: To tally cryptographic library calls, ListDLLs, a utility within Microsoft's Sysinternals toolkit, is employed. This tool facilitates the extraction of library calls made by a process identified by its Process ID, which is obtained from a log file generated in preceding steps. Subsequently, the output is analyzed by juxtaposing it with the standard libraries accessed during cryptomining activities.

### 4.3.3 Data Collection and Sandboxing:

The decision to utilize a sandbox environment stemmed from the recognition of the inherent challenges associated with removing malware once a system has been compromised. By conducting the analysis within a controlled virtual environment, efforts were made to mitigate the risks associated with malware infections while enabling comprehensive analysis and experimentation.

A controlled sandbox environment was established using Oracle Virtual-Box, incorporating a Windows 10 version 22H2 installation. The sandbox was meticulously configured with 8GB of RAM, 128GB of ROM, and allocated 4 CPU cores. This configuration was purposefully chosen to provide ample resources for executing diverse applications and conducting thorough malware analysis tasks efficiently.

To emulate a realistic user environment and enhance the authenticity of our experiments, a suite of commonly used applications such as Firefox, Spotify, VLC, among others, was installed. These applications were selected to mirror the usage patterns of everyday users, thereby effectively simulating real-world scenarios. Additionally, the inclusion of these applications served to counter anti-VM measures commonly employed by malware, ensuring a

more accurate representation of potential threats.

During the construction of our malware dataset, samples were obtained from reputable repositories. Each sample underwent meticulous scrutiny, including manual inspection and analysis, to validate its authenticity and relevance to our study.

To systematically analyze each malware sample, a rigorous testing protocol was implemented. A snapshot of the virtual machine (VM) environment was created before executing each sample. Subsequently, each malware sample was individually executed within the isolated environment. Following execution, feature extraction was conducted to identify the presence of cryptojacking behavior, capturing pertinent features and observations.

To maintain the integrity of the analysis and ensure repeatability, VM was reverted to the pre-execution snapshot after each analysis session. This ensured the elimination of any potential residual effects and prepared the environment for the analysis of subsequent samples.

This systematic approach allowed us to construct a comprehensive dataset comprising verified malware samples exhibiting cryptojacking behavior, facilitating detailed analysis and evaluation of detection and mitigation strategies.

## 4.4 Dynamic Analysis:

The malware is executed within a sandboxed environment for a duration of 40 seconds, and its metrics are retrieved. Based on these metrics, the following three models are implemented.

### **Sequential Analysis:**

The sequential analysis model, inspired from Advanced Behavior-Based Technique for Cryptojacking Malware Detection[7], employs a multi-stage approach, meticulously analyzing each process for signs of malicious activity.

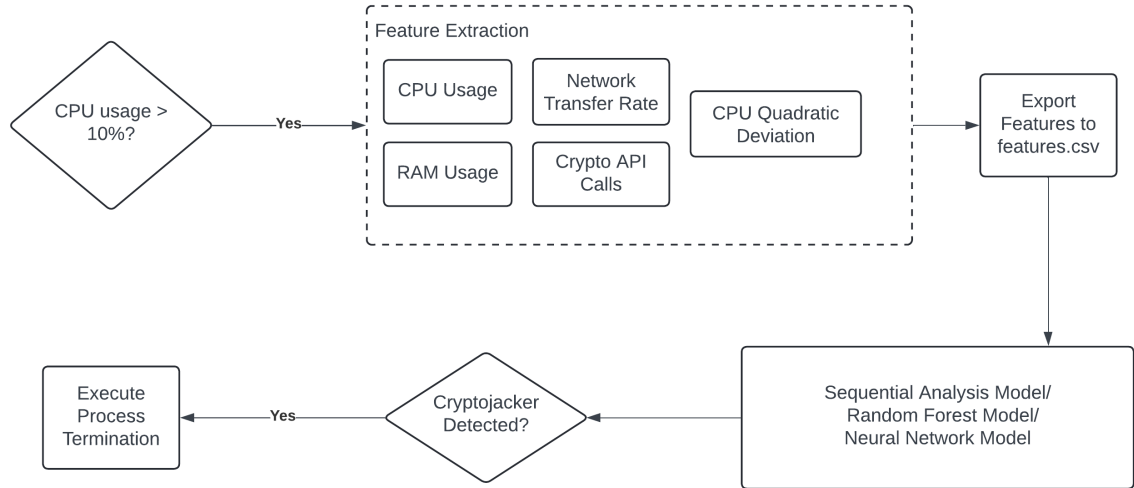


Figure 4.3: Dynamic Analysis Overview

**Stage 1 - Start:** Monitoring of the overall CPU usage of the system commences. If it surpasses a predefined threshold, triggering of analysis occurs, leading to investigation of individual processes contributing to high resource consumption. Each suspicious process undergoes launching in a separate thread for efficient analysis.

**Stage 2 - Process CPU utilization:** Forwarding of the PID to the subsequent analysis stage occurs only if the process exceeds a predefined threshold of 10

**Stage 3 - Network Access:** Monitoring of each suspected process for network access activity is conducted. Detection of a connection within a specific timeframe of 40 seconds raises suspicion.

**Stage 4 - Network Rate:** Monitoring of the network upload rate is performed to check for surpassing of a threshold of 55KB/sec, forwarding PIDs meeting this criterion to the next analysis stage.

**Stage 5 - Average Quadratic Deviation:** Calculation of the Average Quadratic Deviation (AQD) of the process's CPU usage for a defined interval is executed. Processes with an AQD less than 10.5 are whitelisted to prevent false positives.

**Stage 6 - Cryptographic Library Calls:** Processes in stage 8 remain under observation, with continued monitoring of their behavior to detect if two or more cryptographic API calls are made from the seven mentioned in Table 4.1. If so, they are deemed to be cryptojackers and are added to the blacklist.

**Stage 7 - Ram Usage:** Analysis of the RAM usage of each remaining process is conducted. Processes meeting a threshold of 2400 KB are forwarded to the subsequent stage.

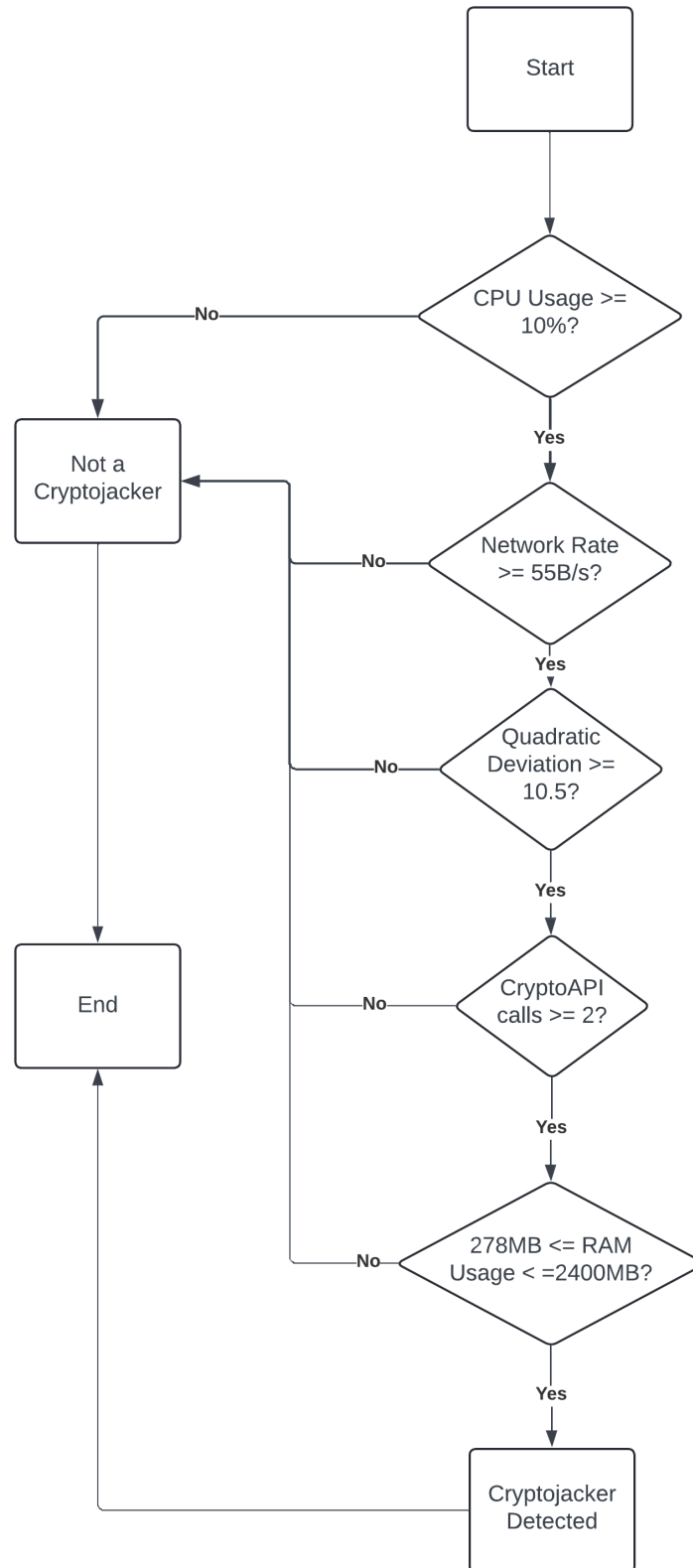


Figure 4.4: Sequential Analysis Overview

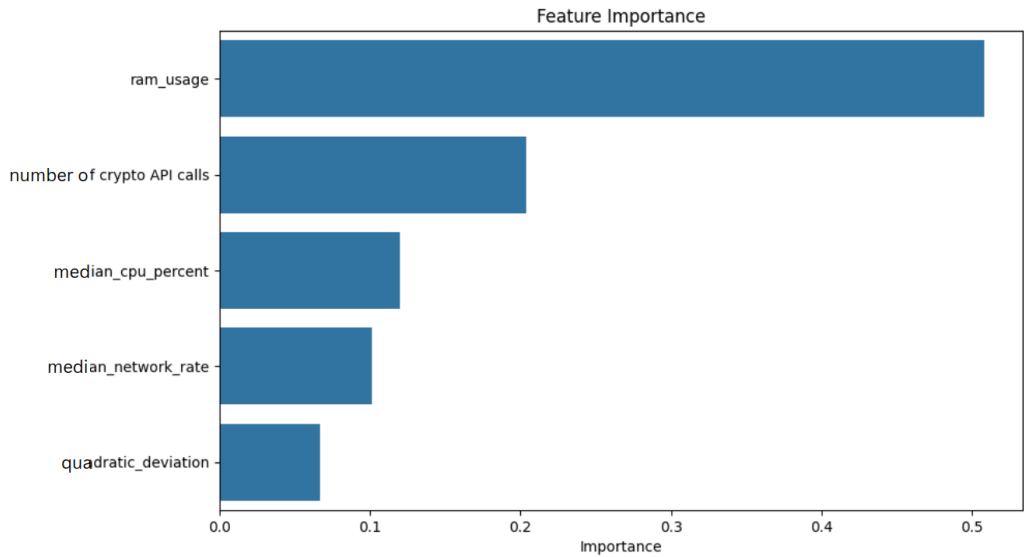


Figure 4.5: Relative importance of features with respect to Random Forest Classification

### Random Forests implementation

The same set of features are employed to train a Random Forest model, which is compared with the sequential analysis model to draw conclusions based on the findings. Random Forest is an ensemble learning technique that constructs multiple decision trees and aggregates their predictions. This approach helps mitigate overfitting and enhances generalization performance compared to individual decision trees. In this study, the model was trained with 435 samples, encompassing both cryptojacking malware and benign software. In the developed tool, upon detection of malware by the Random Forest classifier, the associated process is terminated, and the cycle is repeated for further analysis.



Table 4.2: Classification Report from Random Forest Classification

| Accuracy | Precision | Recall | F1-score |
|----------|-----------|--------|----------|
| 0.99     | 1.0       | 0.98   | 0.99     |

Table 4.3: Confusion Matrix of the Random Forest Classification

| <b>Actual/Predicted</b> | <b>Yes</b> | <b>No</b> |
|-------------------------|------------|-----------|
| Yes                     | 61         | 1         |
| No                      | 0          | 38        |

### Deep Learning implementation

Deep learning is a kind of machine learning that employs neural networks with numerous layers to extract complex patterns from data. These models have exhibited exceptional performance in several areas, such as image recognition, natural language processing, and anomaly detection. Within the scope of this project, advanced deep learning methods is utilized to categorize processes as either cryptojackers or non-cryptojackers, using extracted features as the basis for classification.

The deep learning method utilizes the CSV file generated during the feature extraction phase as input for process categorization. The dataset has been preprocessed using the Pandas package to eliminate unnecessary identifiers and encode the target variable, 'Cryptojacker or not,' as binary values. In addition, the process of feature standardization is carried out using the StandardScaler from the sklearn package. This is done to maintain consistency in the scale of the features, which is essential for achieving optimal performance of the neural network.

To develop the model, a feedforward neural network (FNN) is implemented using TensorFlow's Keras API. The design consists of an input layer containing 64 neurons, a hidden layer including 32 neurons, and an output layer containing a single neuron. The output neuron uses a sigmoid acti-

vation function for binary classification. The model is compiled using the Adam optimizer and binary cross-entropy loss function, with accuracy as the major metric of focus.

The model is trained for 50 epochs with a batch size of 32, utilizing an 80/20 split between training and testing data. We trained the model with 435 samples which consist of cryptojacking malwares and goodwares after which, the model is stored for future use in the testing phase. During the testing phase, the model that was stored is loaded, and the processes are categorized according to the features that have been extracted from the CSV file. Processes categorized as cryptojackers are subsequently detected and terminated in order to minimize any security hazards. We trained the model with 435 samples which consist of cryptojacking malwares and goodwares

Table 4.4: Results from the Deep Learning Model

| Accuracy | Precision | Recall | F1-score |
|----------|-----------|--------|----------|
| 0.97     | 1.00      | 0.95   | 0.99     |

Table 4.5: Confusion Matrix of the Deep Learning Model

| <b>Actual/Predicted</b> | <b>Yes</b> | <b>No</b> |
|-------------------------|------------|-----------|
| Yes                     | 59         | 3         |
| No                      | 0          | 38        |

# Chapter 5

## Results and Comparisons

The tests were performed on a VirtualBox virtual machine with following characteristics: Windows 10 OS, version 22H2, quad-core 3GHz CPU and 8GB RAM.

A test dataset comprising 535 unseen samples of active cryptomalware and benign software was created, with 435 samples used for model training and 100 samples for testing. MinerJAB takes 40 seconds to detect malware once it starts running. This delay is deemed acceptable as running cryptomalware for less than a minute has negligible effects on hardware.

Table 5.1: Comparison of results of previous works on cryptojacking detection with MinerJAB

| Name                             | Method Used         | No. of Samples Used | Accuracy (in %) |
|----------------------------------|---------------------|---------------------|-----------------|
| MinerJAB                         | Sequential Analysis | 100                 | 62              |
| MinerJAB                         | Random Forest       | 100                 | 99              |
| MinerJAB                         | Deep Learning       | 100                 | 97              |
| MINOS[12]                        | CNN                 | 300                 | 98              |
| Behavioral Analysis[6]           | Sequential Analysis | 50                  | 82              |
| Advanced Behavioural Analysis[7] | Sequential Analysis | 100                 | 93              |

Following the analysis, it is evident that the sequential model performs poorly, while the random forest and neural network approaches exhibit ex-

ceptional performance.

While some models with larger samples achieve high accuracy rates, other models such as the behavioral and advanced behavioral models yield lower results compared to MinerJAB's Random Forest implementation.

## Chapter 6

### Conclusions and Future work

This project presents a novel approach to identify cryptojacking malware and enhances existing detection methods. A comprehensive literature review was conducted, examining numerous established methodologies in depth.

The primary distinction between existing solutions and the proposed approach lies in targeting host-based cryptojacking malware, whereas most current solutions focus on web-based cryptojacking malware. Additionally, while many studies have focused on older Windows machines like Windows 7 or earlier, this research specifically addresses the latest malware detected on up-to-date versions of Windows 10 and 11. This ensures that the findings are significantly more accurate and relevant to current needs. Furthermore, a comparative analysis of several models utilizing the same datasets and features is conducted to make a definitive comparison between them, a step not taken in earlier works.

However, as future work, this solution could be extended to encompass other popular operating systems. The implementation is modular, so that it's easy to swap out components like the classification model used according to the needs of each scenario. Further studies could focus on GPU-based malicious miners, an area currently lacking in research. The evolution of cryptojacking malware towards greater stealthiness underscores the need for

advanced sandboxing techniques to detect it effectively. Regular studies are essential as cyberthreats continually evolve, emphasizing the importance of maintaining ongoing awareness and vigilance.

# References

- [1] B. O’Gorman, “Internet security threat report 2019.” <https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf>, 2019.
- [2] K. P. Jayasinghe and Guhanathan, “A survey of attack instances of cryptojacking targeting cloud infrastructure,” *ACM*, pp. 100–107, 2020.
- [3] M. K. Saad, A. Mohaisen, and David, “End-to-end analysis of in-browser cryptojacking,” 2018.
- [4] “Dr.mine.” <https://github.com/1lastBr3ath/drmine>, 2018. Accessed: 2023-10-09.
- [5] “Minerblock.” <https://github.com/xd4rker/MinerBlock>, 2018. Accessed: 2023-10-09.
- [6] D. Tanana, “Behavior-based detection of cryptojacking malware,” in *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT)*, pp. 0543–0545, 2020.
- [7] D. Tanana and G. Tanana, “Advanced behavior-based technique for cryptojacking malware detection,” in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–4, 2020.

- [8] O. Sanda, M. Pavlidis, and N. Polatidis, “A deep learning approach for hostbased cryptojacking malware detection,” *Evolving Systems*, 2023.
- [9] F. Gomes and M. Correia, “Cryptojacking detection with cpu usage metrics,” in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–10, 2020.
- [10] G. Xu, W. Dong, J. Xing, W. Lei, J. Liu, L. Gong, M. Feng, X. Zheng, and S. Liu, “Delay-cj: A novel cryptojacking covert attack method based on delayed strategy and its detection,” *Digital Communications and Networks*, 2022.
- [11] N. Gaidamakin and D. Tanana, “Naïve bayes cryptojacking detector,” in *2022 Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 259–262, 2022.
- [12] F. N. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac, “Minos: A lightweight real-time cryptojacking detection system,” *Proceedings 2021 Network and Distributed System Security Symposium*, 2021.
- [13] G. Mani, M. Kim, B. Bhargava, P. Angin, A. Deniz, and V. Pasumarti, “Malware speaks! deep learning based assembly code processing for detecting evasive cryptojacking,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–17, 2023.





