A Deep Learning Approach for Host-Based Cryptojacking Malware Detection

Abstract

With the continued growth and popularity of blockchain-based cryptocurrencies there is a parallel growth in illegal mining to earn cryptocurrency. Since mining for cryptocurrencies requires high computational resource; malicious actors have resorted to using malicious file downloads and other methods to illegally use a victim's system to mine for cryptocurrency without them knowing. This process is known as host-based cryptojacking and is gradually becoming one of the most popular cyberthreats in recent years. There are some proposed traditional machine learning methods to detect host-based cryptojacking but only a few have proposed using deep-learning models for detection. This paper presents a novel approach, dubbed CryptoJackingModel. This approach is a deep-learning host-based cryptojacking detection model that will effectively detect evolving host-based cryptojacking techniques and reduce false positives and false negatives. The approach has an overall accuracy of 98% on a dataset of 129,380 samples and a low performance overhead making it highly scalable. This approach will be an improvement of current countermeasures for detecting, mitigating, and preventing cryptojacking.

Keywords: Blockchain, Cryptojacking, Deep-Learning, Illegal-Mining, Machine Learning, Crypto Malware Detection.

1. Introduction

The rapid growth of blockchain, especially in the financial (cryptocurrency) sector has been on the rise but a lack of regulatory compliance has created many security vulnerabilities around this emerging technology (Sanda et. al., 2022). Most cryptocurrencies use the Proof-of-Work (PoW) mining process to validate transactions; this process involves advanced mathematical computation that requires powerful systems with high processing power (Tayyab et. al., 2022). This computational difficulty has become more complex over time thereby creating a mining pool epidemic. Malicious actors and cybercriminals are now capitalizing on this vulnerable industry more than ever (Connolly & Wall, 2019). Illegal cryptocurrency mining has transcended the reigning cyberthreat, ransomware as the most persistent cyber-attack of today (Tekiner et. al., 2021). Blockchain-based cryptocurrency applications require high computational resources to generate cryptocurrency. The use of cryptocurrency has surged in recent years, and so has the prevalence of cryptojacking, a form of cyberattack that hijacks a victim's computing resources to mine cryptocurrency (Xu et. al., 2022).

Cryptojacking can be defined as the unlawful use of a person's or company's computer or mobile device by malicious actors to mine for cryptocurrencies like Bitcoin and Monero (Toulas, 2022). A malicious actor can use this attack to increase their computational power which can pose a risk to any blockchain-based on mining process. Malicious actors have resorted to other means by illegally using a host's computer to mine for cryptocurrency and current studies have shown that attackers are pooling the processing power of many mobile phones to mine PoW based cryptocurrencies (Hernandez-Suarez et. al., 2022). This activity can be done in a stealthy manner, causing little noticeable impact on the victim's computer but resulting in significant profits for the attacker; this is known as host-based cryptojacking (Tayyab et. al., 2022).

Despite a drop in cryptocurrency prices, there is a rise in blockchain-based cryptocurrency applications (Xu et. al., 2022). As the use of cryptocurrency continues to grow, host-based cryptojacking attacks are becoming increasingly common, more dynamic, and sophisticated, making it challenging for traditional detection methods to identify them (Aponte-Nova et. al., 2022). While host-based cryptojacking is perceived as a less destructive cyber-threat; this is false. If for example, an electronic medical record (EMR) system is targeted; this could deter staff from being able to promptly access critical patient information due to performance problems caused by the crypto-malware, this could lead to loss of lives and delayed diagnosis (Khan Abbasi et. al., 2023).

Detecting host-based cryptojacking can be challenging since the activity is often designed to be stealthy and to consume computational resources only when the host is idle (Aponte-Novoa et. al., 2022). The critical problem of host-based cryptojacking is the impact it can have on the affected system and its host (users). This can degrade the performance of the CPU or GPU leading to slow system productivity and can be highly impactful on resource constrained environments (Varlioglu et. al., 2022). The security risks and financial losses caused by host-based cryptojacking can be significant. This can lead to data theft, unauthorized access and open the system to other malicious activities/attacks. Cryptojacking is also known to cause damage to the reputation of the organization, especially in the case of security organizations; a data breach of sensitive information will lead to lack of trust from stakeholders.

In the realm of detecting host-based cryptojacking, despite the existence of proposed solutions and studies utilizing machine learning (ML) techniques, there are several challenges that need to be addressed to enhance the effectiveness of ML-based detection. These challenges encompass the limited availability of labeled datasets suitable for training ML models to effectively learn patterns of cryptojacking activity, the ever-evolving and dynamic nature of cryptojacking malware and techniques that can evade detection from ML models trained on historical data, an overreliance on CPU metrics as the primary indicators for detecting cryptojacking attacks, which may not suffice considering the rapid evolution of cryptojacking malware, and the potential occurrence of false positives (identifying non-cryptojacking activity as cryptojacking) and false negatives (failing to detect cryptojacking activity), resulting in unwanted alerts or missed detection. Addressing these challenges is crucial to bolster the accuracy and efficacy of ML-based cryptojacking detection mechanisms.

In order to mitigate the risks posed by host-based cryptojacking, we put forward a deep-learning model named 'CryptoJackingModel'. This model analyses a range of system metrics, including CPU memory usage and process behavior, surpassing the sole reliance on CPU usage and performance metrics. By incorporating these additional factors, our proposed model effectively detects patterns of activity within extensive datasets that are characteristic of cryptojacking malware. Machine learning-based solutions, including traditional machine learning algorithms and deep learning algorithms, have shown promise in detecting host-based cryptojacking activity. These approaches use various techniques to analyze system metrics and identify patterns of activity that are indicative of cryptojacking. To address some of these challenges, this paper uses a published dataset that contains normal behavior of system metrics without cryptojacking malware and abnormal behavior of the system metrics when infected with a cryptojacking malware. We present a deep-learning approach that learns to distinguish and classify between normal and abnormal cryptojacking activity patterns, enabling it to detect host-based cryptojacking activity even in cases where the activity is carefully disguised.

Our proposed CryptoJackingModel solution will effectively detect evolving host-based cryptojacking techniques and reduce false positives and false negatives. Hence, this approach will evaluate the results from the proposed model and compare with traditional methods like KNN, Isolation Forest, decision tree and SVM used in previous works from (Xavier and Sahni, 2020) and (Gomes and Correia, 2020). The studies conducted by Gomes and Correia (2020) and Xavier and Sahni (2020) have employed the identical dataset utilized in this research for binary classification purposes. Our proposed deep-learning model will demonstrate notable advancements over their findings. To evaluate the effectiveness of our proposed deep-learning model we conducted experiments using the features of abnormal and normal CPU behavior and compare the results with traditional machine learning methods.

Overall, our proposed method shows reliable results in detecting host-based cryptojacking activity using deep-learning. Our results show that our proposed method outperforms traditional machine learning methods, achieving high detection rates with low false positive rates. The proposed approach has the potential to support organizations to detect, prevent and mitigate cryptojacking attacks, protecting their computing resources and improving their general cybersecurity posture.

The contributions of this paper are listed as follows:

- Deep-learning approach for distinguishing between normal and abnormal host-based cryptojacking activity patterns: The paper presents a novel deep-learning approach that effectively learns to differentiate between normal and abnormal host-based cryptojacking activity. This approach enables the detection of host-based cryptojacking even in cases where the malicious activity is carefully disguised or obfuscated. By leveraging deep-learning techniques, the proposed model enhances the accuracy and effectiveness of detecting host-based cryptojacking attacks.
- Advancement of the state-of-the-art detection accuracy: The paper achieves a significant advancement in the field of host-based cryptojacking detection by demonstrating a detection accuracy of 98%. This high level of accuracy is achieved through the implementation of the proposed deep-learning model. The paper provides experimental results, analyses, and evaluations to validate and showcase the effectiveness of the model compared to existing solutions. This advancement in detection accuracy contributes to enhancing overall security and protection against host-based cryptojacking threats.

Additionally, the paper demonstrates the use of CPU metrics for analyzing host-based cryptojacking attacks. By utilizing CPU metrics, the proposed model provides insights into the behavior and impact of host-based cryptojacking malware. There are several factors influencing the computational cost like training time, inference time, data preprocessing, model complexity and choice of hardware. This will usually vary depending on the deep learning model and the experimental setup. This analysis further contributes to the understanding and identification of such malicious activities, aiding in the development of effective countermeasures.

The paper is organized as follows: in the next section we discuss background information and the related works. Then we discussed methodology and evaluation in section 3. Section 4 is followed by results and presentations. We conclude with the conclusion section in Section 5.

2. Background Information

In recent years, cryptocurrency mining has become increasingly popular, with numerous cryptocurrencies like Monero and Bitcoin being mined for profit (Varlioglu et. al., 2020). This has led to the emergence of a new type of attack known as "Cryptojacking," where malicious attackers exploit the computing resources of others to mine cryptocurrencies without their knowledge or consent (Saad and Mohaisen, 2018). Cryptojacking attacks can occur on various platforms, including desktop computers, mobile devices, and even Internet of Things (IoT) devices. Host-based cryptojacking detection is an effective approach of detecting these attacks on victims (host) computers. This approach involves analyzing the behavior of the host, such as the CPU usage, network traffic, and system calls, to identify abnormalities that could indicate a cryptojacking attack. Host-based cryptojacking attacks have become increasingly common in recent years, with attackers using a range of techniques to evade detection, such as running the mining software at low intensity, targeting specific CPU cores, and using obfuscation techniques (Sivaraju, 2022). These techniques operate as follows:

Running the mining software at low intensity: this is a technique employed by malicious attackers to evade detection mechanism when performing cryptojacking attacks (Sivaraju, 2022). The attacker intentionally limits the power of the system resource (GPU or CPU) used when illegally mining to avoid triggering any spikes or heavy usage that could trigger detection software or alert the victim (Caprolu et., al. 2019). This technique allows the attack to happen for a long time before it is detected or in some cases they never get detected.

Targeting specific CPU cores: this form of evading detecting targets CPU cores that are the least utilized and this will not have an indication on the overall performance of the system (Gomes et. al., 2020). This will allow the attacker to move stealthily and mine for cryptocurrency for a long time.

Hiding the mining process from the host (victim): the attacker uses rootkits or modifies the registry to avoid any traces of the mining software on the system. The whole point of this technique is to infiltrate parts of the system that can make the mining process a secret (Gomes et. al., 2020).

Obfuscation techniques: the malicious attackers engage in using different obfuscation techniques to avoid detection (Tekiner et. al., 2021). An example is using code obfuscation techniques to make the mining process and software more confusing and difficult to analyze. Some other obfuscation techniques are using fileless malware and anti-debussing techniques.

As a result, detecting host-based cryptojacking activity has become more challenging for traditional methods, such as signature-based detection and rule-based detection (Hernandez-Suarez et. al., 2022).

A cryptojacking malware attack starts with the Script Preparation, followed by the Script Injection then The Attack takes place. The Script Preparation and The Attack phase for all cryptojacking malware types are always the same. The Script Injection phase is usually different, and this can be done in two distinct ways, which are by injecting the malicious malware script into the website or by injecting it into the application. These can be referred to as In-Browser cryptojacking or Host-Based cryptojacking.

In-Browser Cryptojacking: uses the technologies provided by websites to gain unlawful access to the user's system to illegal mine for cryptocurrency.

Host-Based Cryptojacking: Host-based cryptojacking refers to the unauthorized use of a host's (system/server) computing resources to mine cryptocurrency which is the focus of this studies. This can be accomplished through the installation of malicious software or the exploitation of vulnerabilities in different software running on the hosts computer. Once installed, the cryptojacking software can use the host's CPU or GPU to perform complex mathematical calculations required for mining different cryptocurrencies (Petrov et. al., 2020). This type of cryptojacking script injection is not done through a web script but instead must be directly installed on the host system. In this type of attack silent cryptomalware attacks the victim's resources (IT/OT Systems) for illegal crypto mining and must be installed on the host's system for the attack to be successful. They are injected into the system via social engineering techniques, embedding into third-party applications, through system vulnerabilities, and as a pay load in the drive-by-download.

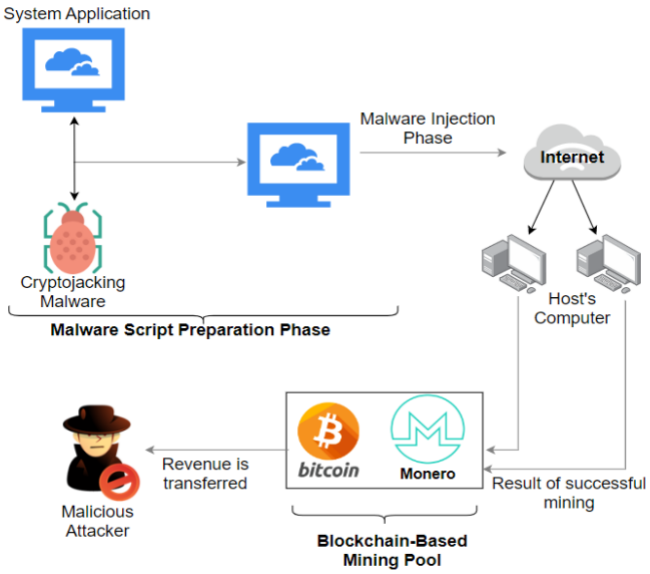

Figure 1: A Visual Representation of Host-Based Cryptojacking Lifecycle

While In-Browser cryptojacking attacks has seen popularity in the past years, there is a rise in host-based cryptojacking attacks (Gomes and Correia, 2020). Sectors like healthcare, government services and education have been major targets. Recently there has been a wave of cryptojacking attacks on retail systems. Figure 1 shows the lifecycle of a host-based cryptojacking attack from the script preparation phase to the script injection phase and the transfer of illegal revenue to the malicious attacker. The malicious attacker merges the cryptojacking script with a trusted application and uploads it into the cloud for the host (victim) to download and install on their operating system (for example, water utility server, healthcare systems or retail servers). This uses the system to mine for cryptocurrency illegally and when this is done the malicious attacker receives the revenue without expending any energy or resources (Romano et. al., 2020).

2.1 Cases of Major Cryptojacking Attacks

In 2018, cryptojacking malware was found in the operation technology (OT) of a water utility company in Europe (Tayyab et. al., 2022). This OT was for monitoring and controlling of the water systems. The cryptojacking malware was able to disable scanners and other defense

mechanisms in the water utility system. Any disruption to industrial systems can cause serious harm and damage. These attacks are due to the increase in blockchain-based applications in the financial sector.

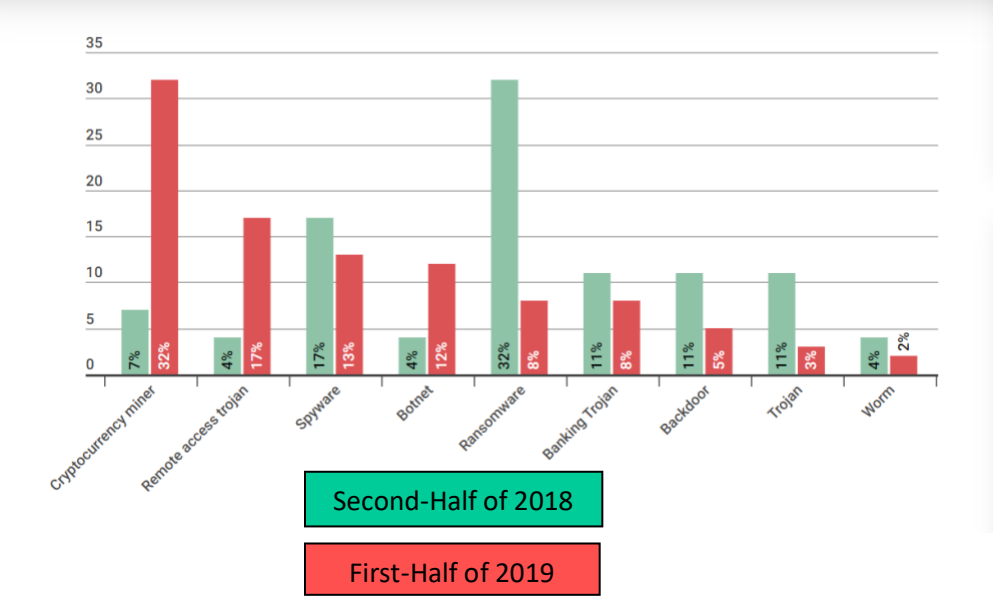

Figure 2: Vulnerability and Threat Trends between the first half of 2018-2019 (Zvelo 2018)

Figure 2 gives a snapshot on how cryptojacking overtook ransomware attacks over a brief period. The chart above is a representation of the dramatic shift in cyber tactics since the widespread development of blockchain-based cryptocurrency applications. The chart shows that cryptomining and ransomware are the two fastest growing malware categories more than other malware categories combined (Skybox Security 2021). According to Skybox report of 2022 more than 70% of organizations have experienced some form of illegal cryptomining. This shows that malicious actors are devising new ways of exploiting the latest weaknesses by creating novel cryptojacking malware to covertly gain control of CPU's and GPU's (Skybox Security 2021).

According to a recent report by Kaspersky, 2022 saw the largest number of users (over half-a-million) affected by malicious cryptojacking malware (Frinconi, 2023). This proves that existing detection methods are inefficient or new mining programs are exploiting unknown vulnerabilities. Between January 2022 to October 2022 Kaspersky's solution detected over 200,000 new modifications of cryptomalware (Frinconi, 2023). It is also key to note that cryptojacking attacks can also lead to crypto-ransomware attacks. This can be disastrous for both individual users and corporate organizations. In a recent survey by Avast, they talk about VenomSoftX that targeted Blockchain.com, Binance, Coinbase, Gate.io, YouTube and Kucoin (Toulas, 2022). It recently cost a service provider over $100,000 for Monero cryptocurrency mining in their service (Toulas, 2022). Detecting and preventing cryptojacking attacks is a major step towards securing the blockchain ecosystem and achieving regulatory compliance. A threat report released by SonicWall threat researchers reported a significant increase in cryptojacking attacks in 2022. This sharp increase has pushed the number of attacks beyond the 100-million

mark, which is the first time this has ever happened. Figure 3 shows the gradually increase in cryptojacking from 2018 to 2022.

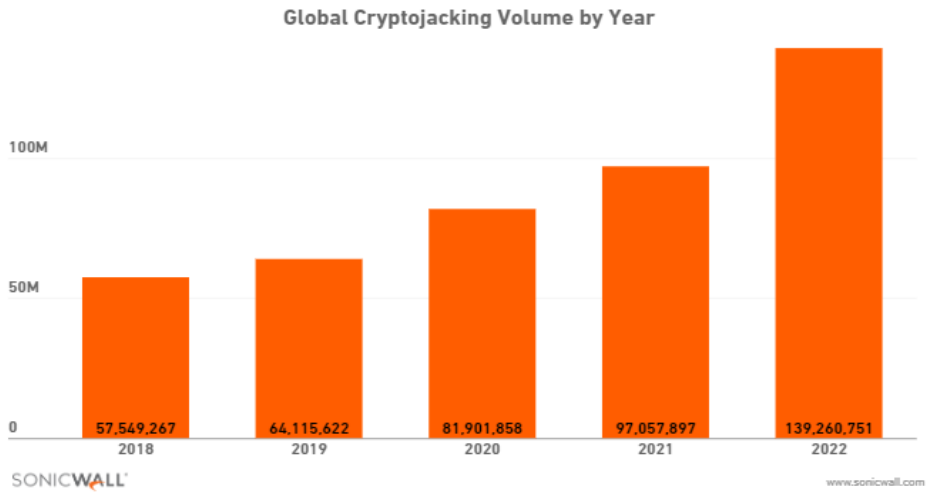

Figure 3: The Global Cryptojacking Volume from 2018-2022 (SonicWall, 2023)

A recent discovery of cryptojacking malware was found in MacOS. This malware makes use of an XMRig tool and was executed alongside the Apple developed video editing software Final Cut Pro (Benyo, 2023). The cryptojacking malware makes use of an invisible internet project (ip2) for communication. This discovery has presented an opportunity to study this new cryptojacking malware and its iterations. Figure 4 below shows the workflow of the cryptojacking malware executing XMRig on the MacOS.

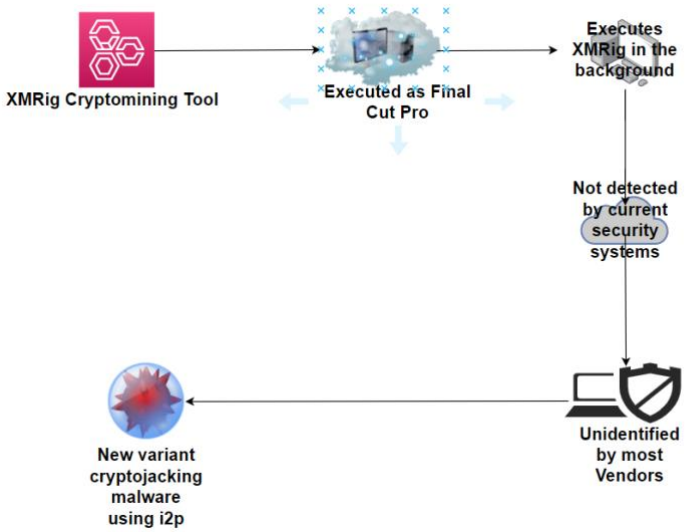

Figure 4: Workflow of the XMRig tool executing on MacOS.

2.2 Significant problems of Cryptojacking:

- Blockchain's cryptomalware abuses their victim's computational power as opposed to infecting, harming, and controlling the system; this makes detection uncertain especially for users with legitimate heavy workloads (Gomes et. al., 2020).
- This poses an elevated risk to any blockchain based on mining as opposed to staking (Aponte-Nova et. al., 2022).
- Security operations/practices that allow the event listed above to occur can cause high maintenance costs and open a larger attack surface for more attacks like sponge attacks and crypto ransomware which can be extremely dangerous for corporate organizations (Gomes et. al., 2020).
- A high percentage of the literature available today addresses mostly browser-based or in-browser mining which limits existing solutions (Caprolu et., al. 2019).
- There is a need to understand the lifecycle of host-based crypto-malware to further secure blockchain-based cryptocurrency applications from theft (Connolly & Wall, 2019).
- The illegal mining of cryptocurrencies (cryptojacking) can have an adverse effect on the blockchain organization in the form of theft and affect users negatively. This activity has created a passive income for malicious attackers making blockchain compliance to existing financial and data regulations difficult (Tekiner et. al., 2021).

Previous works on detecting cryptojacking activity using ML-models has focused on using traditional machine learning algorithms, such as Support Vector Machines (SVM), KNN, and Isolation Forest. However, these methods have limitations in their ability to capture complex patterns and may suffer from overfitting or poor generalization to new data. The detection of cryptojacking activity is complex, as it often operates stealthily and disguises itself within legitimate processes, making it difficult to differentiate from regular system operations. Additionally, the evolving nature of cryptojacking malware presents an ongoing challenge, as attackers constantly adapt their techniques to evade detection and exploit vulnerabilities. Another issue is the limited availability of labeled datasets, hindering the development of accurate machine learning models for cryptojacking detection. Furthermore, the reliance on CPU metrics alone may not be sufficient to detect cryptojacking attacks, necessitating the exploration of alternative indicators and behaviors associated with cryptojacking. Finally, false positives and false negatives in detection can lead to undesirable consequences, including unnecessary alerts or missed detection of actual cryptojacking activity. Addressing these challenges requires a multi-faceted approach, including the development of sophisticated detection mechanisms, the continuous monitoring and analysis of network and system behavior, and the implementation of robust security measures to mitigate the risk of cryptojacking.

2.3 Related Work

Cryptojacking has become a huge threat to cybersecurity as discussed in the background section and host-based cryptojacking is on the incline (Carlin et. al., 2020). There are studies and datasets related to the detection and prevention of cryptojacking attacks with most of the research focused on in-browser cryptojacking and limited studies into host-based cryptojacking. The

literature presents different detection methods for host-based cryptojacking, and these solutions prompt some limitations related to effectiveness, transparency, small datasets, and performance. We will categorize this into three sections (Detection, Mitigation and Case Studies)

2.3.1 Detection Techniques

Techniques for detecting host-based cryptojacking can be grouped under static analysis, dynamic analysis, and hybrid analysis. Static analysis refers to the analysis of an application using rule-based detection and machine-learning detection techniques. Dynamic analysis involves executing cryptojacking malware in a virtual or controlled environment using sandboxing and monitoring system calls. Hybrid analysis involves a combination of static and dynamic analysis to make a more accurate detection of host-based cryptojacking. The studies in this category use one or more of these detection techniques.

In (Caprolu et., al. 2019), they propose an ML based solution that detects cryptocurrency related activities like pool mining and solo mining. Their solution focuses on mining activities that can lead to a sponge attack.

A major step towards cryptojacking detection in IoT devices was made by (Ahmad et. al., 2019) where they proposed a lightweight cryptojacking classifier model. This solution considers the small processing capabilities of IoT devices and chooses an ML solution that will fit into a low processing capability ecosystem.

In (Tekiner et. al., 2021) they conduct an analysis of all the cryptojacking prevention and detection solutions out in the market and propose a systematic study on emerging cryptojacking malware. The work from (Razali & Shariff, 2019) introduces CMBlock which makes use of a web extension for browsers that can detect mining scripts for in-browser cryptojacking. By applying this solution, the application can detect unknown domains that are not blacklisted.

"Machine Learning Approaches to Detect Browser-Based Cryptomining" by Xavier and Sahni, (2020) proposed a machine learning approach to detect browser based cryptojacking by studying the complexity of Javascript codes and performance parameters of a system. While the results are promising, the paper does not explore the effectiveness of this method on host-based systems and their model performed poorly with unsupervised models.

2.3.2 Mitigation Techniques

Mitigating the impact of host-based cryptojacking uses prevention, detection, and response techniques. The studies in this category use one or more of these mitigation techniques.

The recent work from (Tanana, 2020) introduces a classification of cryptojacking and proposes a mitigation and prevention technique based on CPU load that can be applied to both in-browser and host-based cryptojacking. Their research shows success rate of applying decision tree algorithm to detect cryptojacking.

On the other hand, the work from (Lachtar et. al., 2020) proposes a generic method of preventing/detecting cryptojacking irrespective of the type of application. They systematically present a low overhead, cross stack solution that works by studying a limited number of common

instructions in cryptographic hash functions and evaluates their suitability for cryptojacking detection.

The work from (Gangwal et. al., 2020) proposes a solution to respond to cryptojacking malware on the host's machine by utilizing Hardware Performance Counters (HPC). Their solution considers the cryptocurrencies mined by the top ten mining pools and their results show good success rate for the classifier.

2.3.3 Case Studies

This category relates to studies that analyze real-world examples of host-based cryptojacking. This provides more detailed information on the current landscape of host-based detection and mitigation techniques. They analyze the impact on real-world situations and the papers in this category fall under studies done using case-studies.

In (Darabian et. al., 2020) they provide a case study to analyze the potential application of deep learning algorithms to detect cryptojacking malware on a host's system. They perform both a static and dynamic analysis of cryptojacking malware.

Table 1: Current Cryptojacking Malware Detection Techniques

Study	**Type of Cryptojacking**	**Dataset/Data Collection**	**Solution/Classifier**	**Performance**
Caprolu et., al. 2019	In-Browser	N/A	Supervised learning	TRP=92% FRP=0.8%
Ahmad et. al., 2019	Host-based & In-browser.	Mixture and malicious and non-malicious network packages.	Dendritic cell algorithm	N/A
Razali & Shariff, 2019	In-browser	Blacklisted domains	Blacklisting	N/A
Tanana, 2020	Host-based & In-browser.	Forty in-browser and host-based cryptojacking samples.	Decision Tree	TRP:81%
Lachtar et. al., 2020	Host-based & In-browser.	Not publicly available	Matching techniques	FRP: 0.2%
Gangwal et. al., 2020	Host-based & In-browser.	Miner and user behaviors	Random Forest	Recall: 97% Accuracy: 97.5%
Darabian et. al., 2020	Host-based	Not publicly available	Deep learning	F1:98% FRP=0.6%

Table 1 shows some current cryptojacking detection and prevention techniques with their evaluation metrics and results. These studies have identified that the research on cryptojacking is still in the development phase and much of the current work need to focus on host-based cryptojacking attacks; this is increasing with intensity and more users will become vulnerable to this type of attack soon.

2.4 Limitations of Related Work

From an in-depth study of the related works above, some general and specific limitations of the experiments were identified, and they are as follows:

- Transferability of models used in some of the related works may not have the best accuracy when applied in a different environment. This can be due to dataset bias; models trained on a particular dataset may not adapt to other environments due to non-accurate representation of training dataset. Also, different systems and environments will have varying configurations. The continuous evolution of cryptojacking attacks may not allow models capture new patterns when applied in a different environment.
- One of the major limitations in the related work is the lack of robust and diverse training data to train an effective model which usually results in deficient performance when the model is applied to a real-world dataset. This is because building an effective model requires large dataset size, diversity of data and adequate representation of real-world scenarios.
- Most of these studies encountered challenges when trying to identify the relevant features to detect normal and abnormal host-based cryptojacking activities. This can be because of the dynamic nature of cryptojacking attacks because they continually evolve making it difficult to establish a stable set of features that can accurately distinguish between normal and abnormal attacks. Malicious actors adopt strategies to evade detection making it challenging to identify patterns. Features that differentiate between normal and abnormal behavior vary based on the system and environment that is being simulated and analyzed.
- A lack of high computational resources to run experiments with large dataset, time constraints in processing large datasets and scalability issues were some of the limitations discussed by (Gangwal et. al., 2020) and (Tanana, 2020) in their respective papers.

2.5 Datasets collected and used in Cryptomalware Detection

Crypto malware detection depends on large datasets that capture the major characteristics and behaviors of different variations of cryptomalware. These datasets are used to train and evaluate intrusion detection methods and algorithms. The Table 2 below, show some of the available datasets for cryptomalware detection:

Table 2: Datasets used in Cryptomalware Detection

Dataset Name	Description	Source	Size	Features

Anubis (Anjum et. al., 2022)	Contains large-scale dataset of malicious and non-malicious windows executables, including host-based cryptojacking records.	Anubis Project	20,000 samples	Registry modifications, file systems change and dynamic API calls.
Malica (Nappa et. al., 2015)	Contains real-world samples of cryptominers.	Private experiment dataset	10,000 samples	File metadata, behavior patterns, network traffic.
Browserscope Dataset (Aponte-Nova et. al., 2022)	Contains webpages infected with in-browser cryptomalware.	Browserscope	N/A	Network requests and DOM events.
In-Browser malware dataset (IBMD) (Aponte-Nova et. al., 2022)	Captures real-world in-browser malware samples	University of California	20,000 samples	URL patterns, behavior patterns.
EMBER (Anderson and Roth, 2018)	Contains large-scale dataset of PE files	Georgia Tech	1.1 Million samples	Section and header information, byte-level n-grams.
CICIDS2017 (CICDS2017, 2020)	Captures network traffic containing several types of attacks including cryptomalware.	Canadian Institute of Cybersecurity (CIC)	2.8 million samples	Flow and packet level features.
Microsoft Malware Classification (MS Malware)	Malware classification including cryptomalware.	Microsoft	500,000 samples	API calls, file properties.

(Bosco et. al., 2018)				
Drebin (Arp et. al., 2014)	Contains android malware dataset for cryptomalware on mobile devices.	German research center for artificial intelligence	5,000 samples	API calls and permissions.
Contagio (ImpactCyberTrust, 2019)	Contains cryptomalware collected from several sources.	Contagis Malware Dump	N/A	File types, malware family and file hash.

These datasets shown in the table above provide valuable resources for researchers and practitioners to advance the field of cryptomalware detection and develop effective defenses against these malicious threats. In summary, while machine learning methods have shown promise in detecting host-based cryptojacking attacks, several limitations need to be addressed to improve their accuracy and effectiveness. Our proposed CryptoJackingModel is focused on developing a more robust and transferable model that can manage diverse and evolving threats. We make use of a large-scale and diverse training dataset that is a representative of real-world scenarios. By critically evaluating existing literature on cryptojacking, relevant features for detecting host-based cryptojacking were identified and we can run experiments in a resource enabled environment making it easier to run large datasets on our model. The proposed model can be applied to other host-based scenarios and datasets to detect host-based cryptojacking attacks.

3. Methodology

In this paper, we adopt a methodology with the following stages and as depicted in the Figure 5:

Stage 1: Selection of Dataset

Stage 2: Exploratory Data Analysis

Stage 3: Exploration of Deep-Learning Model

Stage 4: Results and Presentations

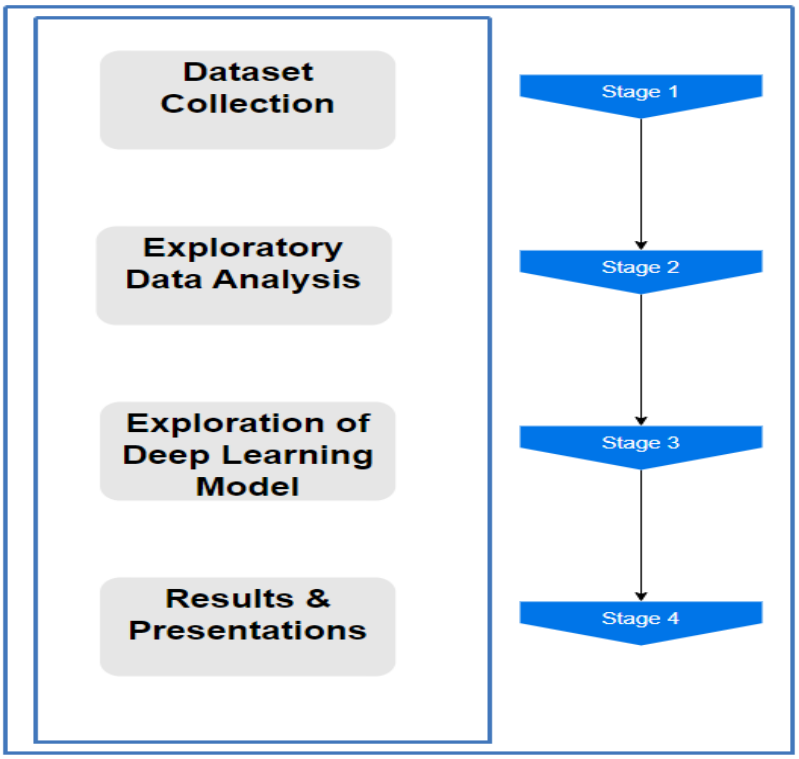

Figure 5: Workflow of Our Proposed Methodology

The development of these stages was done on the Jupiter notebook on Google Colab environment with python programming libraries such as matplotlib, tensorflow, numpy, pandas, Pytorch, and seaborn. We performed our training and execution in the Google Colab virtual environment with 3.41 GHz Intel Core processor. 16GB of RAM and 64-bit operating system.

3.1 Stage 1: Selection of Dataset

We utilize the dataset that was collected by (Barbhuiya et. al., 2018) where they ran lab-based experiments in a lab-based Cloud data center. The cryptojacking attack was simulated using a stress tool to simulate high volume of user traffic on the CPU memory, Input/Output, and disk. Features based on CPU consumption, CPU performance, CPU memory, CPU usage, and CPU time were captured. The collected dataset comprises of 64,690 abnormal entries and 11,558 normal entries. The extracted metrics from the CPU core are shown in Table 3. In addition to these features, we present the Class Labels as Abnormal = 0 and Normal =1.

Table 3: Key features of the cryptojacking dataset

Feature	Description
% Guest	Percentage of total CPU running a virtual processor.

% Idle	Percentage of total CPU that is idle when the system does not have an input and output request.
% Steal	Percentage of total CPU spent in involuntary wait while servicing another processor.
% Soft	Percentage of CPU spent servicing software interrupts.
% Irq	Percentage of CPU servicing hardware interrupts.
% Iowait	Percentage of CPU that is idle during input and output.
% Sys	CPU utilization at system level.
% Nice	CPU usage at user level with good priority.
% User	CPU utilization of host application.
CPU total	Total number of processors installed.
Diskio_sda1_read_bytes	Kbytes recorded per second by disk Sda1.
Fs_/_free	Total kb allowance to be used in the filesystem.
Fs_/_used	Total kb space used in the file system process.
Mem_active	Active memory accessible.
Mem_shared	Total memory shared by the system.
Mem_total	Total available RAM.
Load_CPUcore	Total number of CPU cores on the system.
Mem_available	Total amount of memory available.
Mem_buffers	Storage used for raw disks blocks.
Mem_inactive	Unused memory that has not been accessed.
Mem_used	Current usage of RAM.
Memswap_used	Available swap space for RAM.
Percpu_0_iowait	Total time spent by CPU during a task.
Processcount_total	Total number of processes on the system.
Percpu_0_system	Average performance of the system
Percpu_0_softirq	Average interrupt request on a CPU

According to (Barbhuiya et. al., 2018) who are the original authors and collectors of the dataset, the data was collected by creating a lab-based simulation scenario where a Virtual Machine (VM) hosted in a cloud data center runs a cloud application. At a certain stage, the VM becomes compromised by a cryptojacking attack that consumes its CPU computational power to perform illegal cryptocurrency mining or cryptojacking attack. The dataset is comprised of performance metrics including CPU utilization and network throughput of the virtual machines. The Cloud data center was built in a Lab using Open Stack. Below are some of the assumptions declared when the data was collected.

- **Scenario 1:** Resource utilization of a virtual machine is characterized by some kind of "normal behavior which can be modelled by using ML techniques and methods".

- **Scenario 2:** Cryptojacking attacks consume virtual machine resources significantly and deviates from the normal behavior of the virtual machine resource utilization which can be captured as Abnormal behavior.

The Data Collector module which is responsible for integrating data into a central repository was used to collect CPU utilization metrics of each of the hosted virtual machines (Barbhuiya et. al., 2018). The frequency of collecting the metrics was 5 seconds which allowed capture of the CPU and network usage behavior in detail. The cryptojacking attack was reproduced by running a CPU stress tool (designed to put a computer's central processing unit under heavy and sustained workload to assess its performance, stability, and temperature under high stress conditions) that consumes almost 100% CPU computational power of the virtual machine. This closely relates to real-world cryptojacking attacks where CPU computational power is consumed significantly to perform mining. Considering the scenarios stated above the experimental period was split into:

- Normal period: first five minutes without attack
- Abnormal period: last five minutes under cryptojacking attacks

Additional artificial workload spikes were injected by running the CPU stress tool for instantaneous periods of time consecutively. This produced the dataset with Abnormal and Normal behavior traces which can be used to build and train a learning model. Records with certain number of flags are classified as Abnormal or Malicious and can be used to evaluate host-based cryptojacking attacks. The rows that display suspicious behavior, meaning that they are statistically irregular with the remaining data are flagged as Abnormal or Malicious during the data collection process as discussed by (Barbhuiya et. al., 2018) in their paper. The Negative class data (Abnormal) was generated by simulating an attack with a CPU Stress tool and creates the assumption that cryptojacking attacks will consume significant CPU power in a constant manner.

3.2 Stage 2: Exploratory Data Analysis

This part of the methodology describes the balancing, cleaning, scaling, and preparation of the dataset.

3.2.1 Balancing the Dataset

The dataset was formatted from an imbalanced dataset to a balanced dataset using python libraries and an excel spreadsheet. Pandas and numpy were used to manipulate the data, while scikit-learn was used for resampling the imbalanced dataset. The imbalanced `'cryptojackingfile.csv'` is loaded into python and we check for missing cells using the print`(data.isnull().sum())`function. We use the interpolation method `(data = data.fillna(data.mean())`to fill out any missing cells. Then the imbalanced dataset is resampled using over-sampling techniques to increase the number of minority samples. The imbalanced dataset before oversampling contains (Normal-1 Class (11,558) and (Abnormal-0 Class 64,690). We use over-sampling technique to increase the majority class using SMOTE (Synthetic Minority Over-sampling Technique) algorithm from the imblearn library. We define the features and target variables in the cryptojacking dataset. A fit resample method to the features

and target variables oversamples the majority class using SMOTE. We combine the resampled features and variables in a new dataset.

The output counter from oversampling is now {0: 64,690 1: 64,690}. The resampled balanced dataset was exported to excel, and the VALUE function was used to format the dataset.

3.2.2 Cleaning, Scaling and Preparing the Dataset

The cryptojacking.csv file is split into training and testing set using the train_test_split () function from scikit-learn to split our dataset into training and testing sets. Then we set the target variable as 'Label.' We then proceed with the next steps of creating the neural network, defining the loss function and optimizer, and training the model. We split the test data into a new test set and a validation set. 70% of the data is used for training and 30% is used for validation. We then used the new test set to evaluate the performance of our final model. The output will show the shapes of the new test and validation sets.

We use the `StandardScaler()` to normalize the data. Normalization will help in preprocessing the data to bring all the features to the same scale. This is important because deep learning models are sensitive to the scale of the input features. The `scaler.transform()` method is used to scale the test data based on the mean and standard deviation learned from the training data. This way, we can prevent data leakage and ensure that the test data is processed in the same way as the training data. We then also use the `transform()` method to transform validation data `X_valid` using the mean and standard deviation computed from the training data. It is important to note that we did not fit the validation data to the scaler as this would result in data leakage, where information from the validation set leaks into the training process. Finally, we scale the test data `X_test_data` using `fit_transform()`. We use only the `transform()` method to scale the test data because the mean and standard deviation is based on the training data only.

3.3 Stage 3: Exploration of our proposed Deep-Learning Model

We set some hyperparameters for our deep-learning model. Hyperparameters are parameters that are set before training the model, and they can have a significant impact on the performance of the model. We set the number of epochs to `EPOCHS=50` and train for multiple epochs to improve performance. We set the batch size to `BATCH_SIZE=32 with LEARNING_RATE = 0.00001` due to speed and memory of CPU. This will control how much the weights of the model are updated during each training iteration. Sometimes using a larger learning rate can lead to faster convergence, but it may cause the model to overshoot the optimal weights and perform poorly. We define a custom Dataset class for the training, validation, and test data.

We define a custom neural network model called `CryptoJackingModel` which is a custom neural network model for cryptojacking detection. The `CryptoJackingModel` class defines a neural network with 5 hidden layers and 1 output layer. Each hidden layer has a `Linear` module followed by a `ReLU` activation function, except for the last hidden layer which has a `Linear` module and no activation function. The `dropout` layer is used for regularization by randomly dropping out a fraction of the input units during training. The `forward` method takes in an input tensor `inputs` and passes it through the layers of the neural network, returning the output tensor `x`. We also define the loss function as `nn.BCEWithLogitsLoss()`, which is the binary cross-entropy loss function

that combines a sigmoid activation function and binary cross-entropy loss in a single module. This is useful for our binary classification tasks ‘Normal’ and ‘Abnormal.’ We define the optimizer to be used during training as `torch.optim.SGD`, which is the stochastic gradient descent optimizer with a specified learning rate. The optimizer will update the model parameters based on the gradients of the loss function during training.

The architecture for our `CryptoJackingModel` is shown in the figure 6 below:

```
CryptoJackingModel(
  (layer_1): Linear(in_features=64, out_features=256, bias=True)
  (layer_2): Linear(in_features=256, out_features=128, bias=True)
  (layer_3): Linear(in_features=128, out_features=64, bias=True)
  (layer_4): Linear(in_features=64, out_features=32, bias=True)
  (layer_5): Linear(in_features=32, out_features=16, bias=True)
  (layer_out): Linear(in_features=16, out_features=1, bias=True)
  (relu): ReLU()
  (dropout): Dropout(p=0.1, inplace=False)
)
```

Figure 6: A Snapshot of the CryptoJackingModel Architecture

4 Experiments and Results

We present Stage 4 of our methodology, which encompasses the execution of experiments and the analysis of results obtained using our CryptoJackingModel. The accuracy of our model's predictions on binary classification problems is evaluated using the binary_acc function. To train the CryptoJackingModel, we iterate over the specified number of epochs defined in the EPOCHS variable and update the model's parameters using a stochastic gradient descent optimizer. Within each epoch, the CryptoJackingModel iterates through the training data, performing forward and backward passes to compute the loss and gradients. The model's parameters are then updated using the optimizer.step() function. Throughout this process, we monitor and record the training loss and accuracy for the respective epoch. Subsequently, the model is evaluated on the validation data, and the loss and accuracy on the validation set are computed. Similarly, the validation loss and accuracy for each epoch are tracked. Finally, the training loss, training accuracy, validation loss, and validation accuracy for each epoch are printed, as depicted in Figure 7.

```
Epoch 001: | Train Loss: 0.69432 | Train Acc: 49.979 | Val Loss: 0.69325 | Val Acc: 50.025
Epoch 002: | Train Loss: 0.69233 | Train Acc: 49.981 | Val Loss: 0.69131 | Val Acc: 50.025
Epoch 003: | Train Loss: 0.69038 | Train Acc: 49.989 | Val Loss: 0.68935 | Val Acc: 50.025
Epoch 004: | Train Loss: 0.68836 | Train Acc: 49.984 | Val Loss: 0.68724 | Val Acc: 50.025
Epoch 005: | Train Loss: 0.68615 | Train Acc: 49.974 | Val Loss: 0.68492 | Val Acc: 50.025
Epoch 006: | Train Loss: 0.68372 | Train Acc: 49.984 | Val Loss: 0.68240 | Val Acc: 50.025
Epoch 007: | Train Loss: 0.68104 | Train Acc: 49.982 | Val Loss: 0.67949 | Val Acc: 50.025
Epoch 008: | Train Loss: 0.67782 | Train Acc: 49.988 | Val Loss: 0.67603 | Val Acc: 50.025
Epoch 009: | Train Loss: 0.67404 | Train Acc: 50.698 | Val Loss: 0.67189 | Val Acc: 50.052
Epoch 010: | Train Loss: 0.66952 | Train Acc: 63.103 | Val Loss: 0.66687 | Val Acc: 68.223
Epoch 011: | Train Loss: 0.66393 | Train Acc: 72.314 | Val Loss: 0.66068 | Val Acc: 74.713
Epoch 012: | Train Loss: 0.65699 | Train Acc: 82.377 | Val Loss: 0.65290 | Val Acc: 88.618
Epoch 013: | Train Loss: 0.64820 | Train Acc: 91.861 | Val Loss: 0.64295 | Val Acc: 98.067
```

Figure 7: Our Deep-Learning Model print out for training loss, training accuracy, validation loss, and validation accuracy for each epoch.

The results show that our CryptoJackingModel is improving as the training progresses. The training loss and accuracy are improving, and the validation loss and accuracy are also improving, which is a good sign as shown in Figure 8 and 9. We address overfitting by using regularization techniques such as dropout regularization.

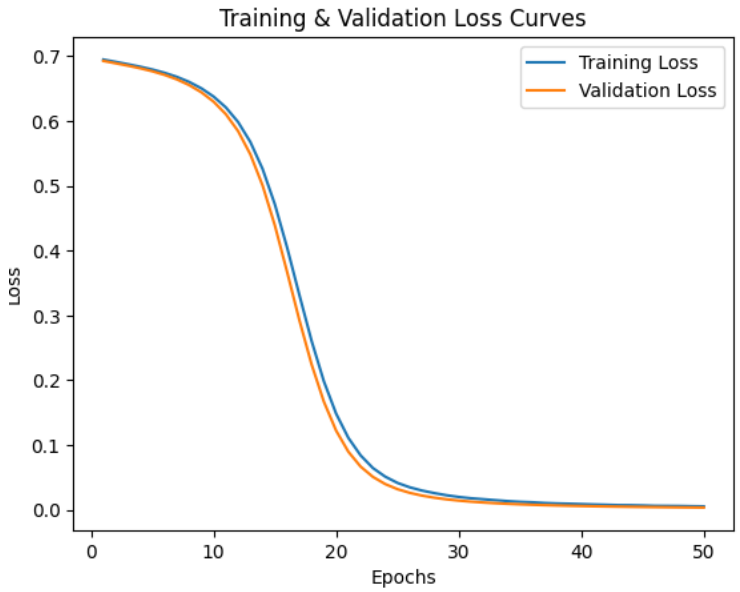

Figure 8: Plot for the Training and Validation Loss Curves.

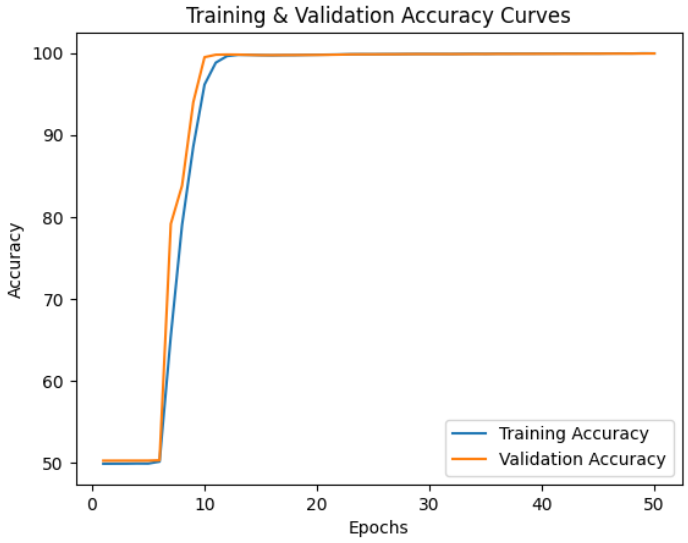

Figure 9: Plot for Training and Validation Accuracy Curves.

Table 4 below show our proposed performance metrics. The target class 0 is 'Abnormal' which is cryptojacking activities and it represent TP for true positive and FP for false negative. The non-target class 1 which is 'Normal', and it represents non-cryptojacking with TN for true negative and FN for false negative, respectively.

Table 4: Proposed Performance Metrics

Evaluation Metrics	Description	Equation
Precision	This represents the correct prediction of classes and measures the standard of the model.	$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}}$ (1)
F1	It is the harmony between recall and precision and measures performance.	$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ (2)
Recall	This represents the average between the classified predicted classes and the whole class.	$Recall = \frac{TP}{TP + FN}$ (3)

Figure 10 shows the confusion matrix with shape (2, 2) with the number of true positives, false positives, false negatives, and true negatives, and Figure 11 shows the classification report respectively.

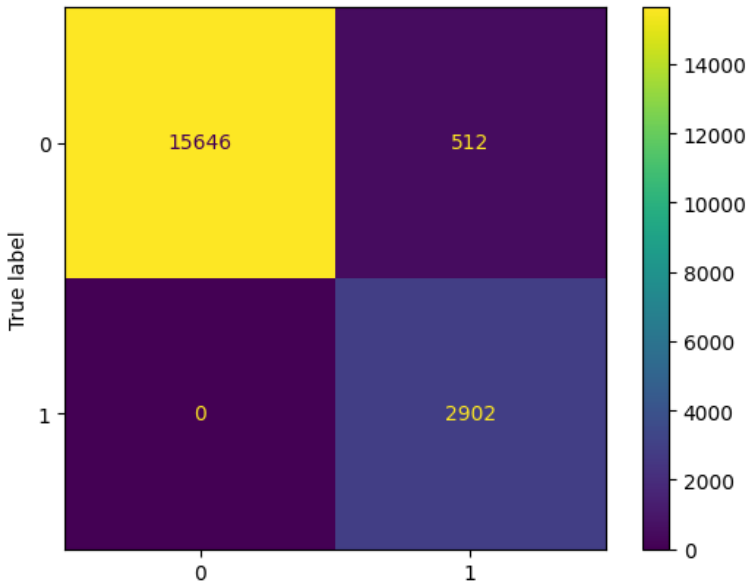

Figure 10: Confusion Matrix to evaluate our proposed Deep-Learning model.

The table 5 shows the classification report evaluating the CryptoJackingModel using precision, recall, and F1. In this case, the CryptoJackingModel gives an accuracy 98%.

Table 5: CryptoJackingModel Evaluation Results

	Accuracy %	**Precision** %	**Recall** %	**F1** %
Run 1	98%	95%	99%	97%
Run 2	97%	93%	98%	95%
Run 3	98%	98%	97%	97%
Average	97.6%	95.3%	98.0%	96.3%

Overall, the classification report shows that the model is performing very well on the test data, with high accuracy and precision, recall, and F1-scores for both classes.

Table 6 presents a comparison of the performance results between the proposed CryptoJackingModel and different classifiers in detecting cryptojacking attacks (Xavier and Sahni, 2020).

Table 6: Comparison with other classifiers

	CryptoJackingModel	**SVM**	**Isolation Forest**	**Local Outlier Factor**
Accuracy	97.6%	78.9%	52.6%	63.2%
Precision	95.3%	66.7%	0%	100%

Recall	98.0%	100%	0%	12.5%
F1	96.3%	80%	0%	22.2%

CryptoJackingModel achieved the highest accuracy rate of 98%, indicating that it correctly classified 98% of the instances in the dataset. SVM, Isolation Forest and Local Outlier Factor achieved an accuracy rate of 78.9%, 52.6%, and 63.2% respectively which is significantly lower than the CryptoJackingModel. Based on these results, it is evident that the proposed CryptoJackingModel outperformed the other classifiers in terms of accuracy, precision, recall, and F1 score. SVM showed moderate performance, while Isolation Forest and Local Outlier Factor performed poorly, particularly in identifying positive instances. These findings suggest that the proposed CryptoJackingModel is a promising classifier for detecting host-based cryptojacking attacks, demonstrating high accuracy and precision with minimal false negatives.

Detecting host-based cryptojacking using deep learning offers several advantages and disadvantages in comparison to the state of the art. The main pros of using deep learning for host-based cryptojacking detection are:

- **Improved accuracy**: Deep learning models have the potential to achieve high accuracy in detecting host-based cryptojacking activities. With their ability to learn complex patterns and features from large datasets, deep learning models can provide more precise identification of cryptojacking behaviors.
- **Adaptability to evolving threats**: Deep learning models can adapt and learn from new and evolving cryptojacking techniques and malware. This flexibility allows the models to stay effective even as cryptojacking methods change over time.
- **Automatic feature extraction**: Deep learning models can automatically learn relevant features from raw input data, eliminating the need for manual feature engineering. This can save time and effort in the detection process.

Despite these advantages, there are also some cons to consider:

- **Large amounts of labeled data**: Deep learning models often require a substantial amount of labeled data for training. Acquiring and labeling such datasets for host-based cryptojacking can be challenging, as it involves identifying and categorizing various cryptojacking activities accurately.
- **Computational resources**: Training deep learning models can be computationally intensive and may require substantial resources, including high-performance hardware and significant training time. This can be a limitation for organizations with limited computational capabilities.
- **False positives and false negatives**: Deep learning models are not immune to false positives (detecting non-cryptojacking activities as cryptojacking) and false negatives (failing to detect cryptojacking activities). Fine-tuning the model and addressing the imbalances in the dataset are necessary to mitigate these issues.

The proposed technique for detecting host-based cryptojacking using deep learning has the potential to be expanded to a wider scientific area through several approaches like: Incorporating diverse and representative datasets from different domains can enhance the technique's coverage,

enabling it to detect host-based cryptojacking activities in a broader range of environments. Additionally, exploring and incorporating domain-specific indicators or patterns as additional features can extend the technique's applicability to different scientific areas. Leveraging transfer learning by fine-tuning pre-trained models on specific scientific domains can also facilitate the adaptation of the technique to new areas, minimizing the need for extensive training on new datasets. Collaboration with experts from various scientific fields can further refine the technique and tailor it to specific industry requirements. Continuous monitoring and adaptation based on feedback and emerging trends in different scientific areas are crucial for ensuring the technique's accuracy and effectiveness. By implementing these approaches, the proposed technique using deep learning can be extended to cover a wider scientific area, enabling the detection of host-based cryptojacking across diverse industries and domains.

Overall, deep learning shows promise in improving the detection of host-based cryptojacking. However, careful consideration must be given to the availability of labeled data, computational resources, and addressing the challenges associated with false positives and false negatives to ensure effective and reliable detection in real-world scenarios.

5. Conclusion

In conclusion, Blockchain Proof-of-Work (PoW) mining process has created an ecosystem of vulnerabilities and attacks on blockchain-based cryptocurrencies (Naseem et, al., 2021). Detecting host-based cryptojacking attacks is a crucial aspect of cybersecurity, given the potential impact on system performance, security, and finances. The use of deep-learning neural networks has emerged as an effective approach to detecting host-based cryptojacking attacks, given their ability to learn complex patterns and features from large datasets.

This paper proposed a deep-learning model to detect cryptojacking attacks focusing on hosts CPU features and host-samples. Current measures like anti-malware, blacklisting and traditional ML-methods have been inadequate to detect host-based cryptojacking attacks. The results from our proposed CryptoJackingModel using deep learning shows that it can manage cryptojacking when compared to traditional machine learning methods. Our study shows that a deep-learning neural network-based approach can achieve high accuracy in detecting host-based cryptojacking attacks. Specifically, our approach achieved an accuracy of 98%, indicating a high degree of confidence in the detection of host-based cryptojacking attacks. This level of accuracy is important as it reduces the chances of false positives and false negatives, which can lead to inefficient use of system resources and potential security and data breaches. The high accuracy of our approach can be attributed to several factors, including the use of a large dataset, effective data exploratory techniques, and the ability of our deep-learning neural network model to learn complex patterns and features.

Overall, our study demonstrates the effectiveness of deep-learning neural network-based approaches in detecting host-based cryptojacking attacks. The CryptoJackingModel can be scaled to be applied to a much larger and balanced dataset; this will allow researchers in the future to take host-based cryptojacking detection further. One of the major limitations was the imbalanced dataset and lack of good samples size, this steered us to apply SMOTE oversampling techniques to get a balanced dataset for easier training of the model While there are still some limitations and

challenges to be addressed, such as the need for enormous amounts of training data and the risk of adversarial attacks, the high accuracy achieved in our study provides a promising foundation for further research and development in this area.

In summary, our study shows the importance of using advanced machine learning techniques to address the emerging threat of host-based cryptojacking attacks. With the increasing sophistication and prevalence of host-based cryptojacking, it is important to develop effective detection and mitigation strategies to minimize their impact on system performance, security, and finances. The proposed methods can be suitable for applicability in other domains because the underlying principles and techniques of deep learning can be transferable to other domains with similar anomaly detection or security-related tasks. However, the suitability of the method for other domains would depend on factors such as the availability of domain-specific data, the need for customization or adaptation to domain-specific characteristics, collaboration with domain experts for validation and fine-tuning, and performance evaluation in the target domain. Further research and evaluation would be necessary to assess the applicability and effectiveness of the proposed method in different domains.

Data Availability

The dataset generated and analyzed during the current study is available from the corresponding author upon reasonable request.

Conflict of Interest

The authors declare that they have no conflict of interest.

References:

Ahmad, A., Shafiuddin, W., Kama, M. N., & Saudi, M. M. (2019). A New Cryptojacking Malware Classifier Model Based on Dendritic Cell Algorithm. ACM International Conference Proceeding Series, 0–4. https://doi.org/10.1145/3387168.3387218

Anderson, H. S., & Roth, P. (2018). *EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models*. http://arxiv.org/abs/1804.04637

Anjum, M. M., Iqbal, S., & Hamelin, B. (2022). ANUBIS: A Provenance Graph-Based Framework for Advanced Persistent Threat Detection. *Proceedings of the ACM Symposium on Applied Computing*, 1684–1693. https://doi.org/10.1145/3477314.3507097

Aponte-Novoa, F. A., Povedano Álvarez, D., Villanueva-Polanco, R., Sandoval Orozco, A. L., & García Villalba, L. J. (2022). On Detecting Cryptojacking on Websites: Revisiting the Use of Classifiers. Sensors, 22(23), 1–15. https://doi.org/10.3390/s22239219

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., & Rieck, K. (2014). *Drebin: Effective and Explainable Detection of Android Malware in Your Pocket*. http://dx.doi.org/

Bosco, Francesca., Shalaginov, Andrii., & Office for Harmonization in the Internal Market (Trade Marks and Designs) (2018). (n.d.). *Identification and analysis of malware on selected suspected copyright-infringing websites.*

Barbhuiya, S., Papazachos, Z., Kilpatrick, P., & Nikolopoulos, D. S. (2018). RADS: Real-time Anomaly Detection System for Cloud Data Centres. 1–14. http://arxiv.org/abs/1811.04481

Benyo M. (2023) Evasive cryptojacking malware targeting macOS found lurking in pirated applications. https://www.jamf.com/blog/cryptojacking-macos-malware-discovered-by-jamf-threat-labs/. Last Accessed on the 3rd March 2023.

Bernstein, L. (1997), 2022 Sonicwall Cyber Threat Report. 5(2), 105–107.https://www.infopoint-security.de/media/2022-sonicwall-cyber-threat-report.pdf

Caprolu, M., Raponi, S., Oligeri, G., & Di Pietro, R. (2019). Cryptomining Makes Noise: A Machine Learning Approach for Cryptojacking Detection. https://doi.org/10.1016/j.comcom.2021.02.016

Carlin, D., Burgess, J., O'Kane, P., & Sezer, S. (2020). You Could Be Mine(d): The Rise of Cryptojacking. IEEE Security and Privacy, 18(2), 16–22. https://doi.org/10.1109/MSEC.2019.2920585

CICDS2017 (2020) "Intrusion Detection Evaluation Dataset" Available at: https://www.kaggle.com/datasets/cicdataset/cicids2017/code. Last Accessed 16th May 2023.

Connolly, L., & Wall, D. S. (2019). The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. Computers and Security, 87(July). https://doi.org/10.1016/j.cose.2019.101568

Darabian, H., Homayounoot, S., Dehghantanha, A., Hashemi, S., Karimipour, H., Parizi, R. M., & Choo, K. K. R. (2020). Detecting Cryptomining Malware: a Deep Learning Approach for Static and Dynamic Analysis. Journal of Grid Computing, 18(2), 293–303. https://doi.org/10.1007/s10723-020-09510-6

Frinconi, P. (2023). The state of cryptojacking in the first three quarters of 2022. https://securelist.com/cryptojacking-report-2022/107898/. Last Accessed: 23rd January 2023.

Gangwal, A., Piazzetta, S. G., Lain, G., & Conti, M. (2020). Detecting covert cryptomining using HPC. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12579 LNCS, 344–364. https://doi.org/10.1007/978-3-030-65411-5_17

Gomes, F., & Correia, M. (2020). Cryptojacking Detection with CPU Usage Metrics. 2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020. https://doi.org/10.1109/NCA51143.2020.9306696

Gomes, G., Dias, L., & Correia, M. (2020). CryingJackpot: Network Flows and Performance Counters against Cryptojacking. 2020 IEEE 19th International Symposium on Network Computing and Applications, NCA 2020. https://doi.org/10.1109/NCA51143.2020.9306698

Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L. K., Olivares-Mercado, J., Portillo-Portilo, J., Avalos, J. G., & Villalba, L. J. G. (2022). Detecting Cryptojacking Web Threats: An Approach with Autoencoders and Deep Dense Neural Networks. Applied Sciences (Switzerland), 12(7). https://doi.org/10.3390/app12073234

ImpactCyberTrust (2019) "Contagio Malware Dump" Available at: https://www.impactcybertrust.org/dataset_view?idDataset=1273. Last Accessed 16th of May 2023.

Jayasinghe, K., & Poravi, G. (2020). A Survey of Attack Instances of Cryptojacking Targeting Cloud Infrastructure. ACM International Conference Proceeding Series, 115, 100–107. https://doi.org/10.1145/3379310.3379323

Khan Abbasi, M. H., Ullah, S., Ahmad, T., & Buriro, A. (2023). A Real-Time Hybrid Approach to Combat In-Browser Cryptojacking Malware. Applied Sciences (Switzerland), 13(4). https://doi.org/10.3390/app13042039

Lachtar, N., Elkhail, A. A., Bacha, A., & Malik, H. (2020). A Cross-Stack Approach towards Defending against Cryptojacking. IEEE Computer Architecture Letters, 19(2), 126–129. https://doi.org/10.1109/LCA.2020.3017457

Nappa, A., Rafique, M. Z., & Caballero, J. (2015). The MALICIA dataset: identification and analysis of drive-by download operations. *International Journal of Information Security*, *14*(1), 15–33. https://doi.org/10.1007/s10207-014-0248-7

Naseem, F., Aris, A., Babun, L., Tekiner, E., & Uluagac, A. S. (2021). MINOS: A Lightweight Real-Time Cryptojacking Detection System. Proceedings 2021 Network and Distributed System Security Symposium (NDSS), February, 1–15. https://doi.org/10.14722/ndss.2021.24444

Norman Xavier, S., & Sahni, V. (2020). Machine Learning Approaches to Detect Browser-Based Cryptomining MSc Internship MSc in Cyber Security Machine Learning Approaches to Detect Browser-Based Cryptomining. https://www.cyberthreatalliance.org/wp-content/uploads/2018/09/CTA-Illicit-CryptoMining-

Petrov, I., Invernizzi, L., & Bursztein, E. (2020). CoinPolice:Detecting Hidden Cryptojacking Attacks with Neural Networks. http://arxiv.org/abs/2006.10861

Romano, A., Zheng, Y., & Wang, W. (2020). MinerRay: Semantics-Aware Analysis for Ever-Evolving Cryptojacking Detection. Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, 1129–1140. https://doi.org/10.1145/3324884.3416580

Razali, M. A., & Mohd Shariff, S. (2019). CMblock: In-browser detection and prevention cryptojacking tool using blacklist and behavior-based detection method. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11870 LNCS(October 2019), 404–414. https://doi.org/10.1007/978-3-030-34032-2_36

Saad, M., Khormali, A., & Mohaisen, A. (2018). End-to-End Analysis of In-Browser Cryptojacking. http://arxiv.org/abs/1809.02152

Sanda, O., & Pavlidis, M. & Polatidis, N., (2022). A Regulatory Readiness Assessment Framework for Blockchain Adoption in Healthcare. 65–87.

Sivaraju, S. S. (2022). An Insight into Deep Learning based Cryptojacking Detection Model. Journal of Trends in Computer Science and Smart Technology, 4(3), 175–184. https://doi.org/10.36548/jtcsst.2022.3.006

SonicWall (2023) "Latest Threat Intelligence Reveals Rising Tide of Cryptojacking" Available at: Latest Threat Intelligence Reveals Rising Tide of Cryptojacking (Accessed 6 April 2023).

Skybox security (2021) Cryptomining is hottest new malware type, research reveals. https://www.skyboxsecurity.com/blog/cryptomining-hottest-new-malware-type-research-reveals/ Last Accessed 17th Febuary 2023.

Tanana, D. (2020). Behavior-Based Detection of Cryptojacking Malware. Proceedings - 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, USBEREIT 2020, 543–545. https://doi.org/10.1109/USBEREIT48449.2020.9117732

Tayyab, U.-H., Khan, F. B., Durad, M. H., Khan, A., & Lee, Y. S. (2022). A Survey of the Recent Trends in Deep Learning Based Malware Detection. Journal of Cybersecurity and Privacy, 2(4), 800–829. https://doi.org/10.3390/jcp2040041

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

Toulas, B. (2022). Google Chrome extension used to steal cryptocurrency and passwords. https://www.bleepingcomputer.com/news/security/google-chrome-extension-used-to-steal-cryptocurrency-passwords/. Last Accessed: 23rd January 2023.

Varlioglu, S., Gonen, B., Ozer, M., & Bastug, M. (2020). Is cryptojacking dead after coinhive shutdown? Proceedings - 3rd International Conference on Information and Computer Technologies, ICICT 2020, 385–389. https://doi.org/10.1109/ICICT50521.2020.00068

Varlioglu, S., Elsayed, N., Elsayed, Z., & Ozer, M. (2022). The Dangerous Combo: Fileless Malware and Cryptojacking. Conference Proceedings - IEEE SOUTHEASTCON, 2022-March, 125–132. https://doi.org/10.1109/SoutheastCon48659.2022.9764043

Xu, G., Dong, W., Xing, J., Lei, W., Liu, J., Gong, L., Feng, M., Zheng, X., & Liu, S. (2022). Delay-CJ: A novel cryptojacking covert attack method based on delayed strategy and its detection. Digital Communications and Networks. https://doi.org/10.1016/j.dcan.2022.04.030

Zvelo (2018) 'Cryptojacking Infection Methods: Identification and Prevention Tips' Available at: https://zvelo.com/cryptojacking-infection-methods-identification-prevention-tips/ (Last Accessed: 15th June 2023).