Cryptojacking detection using local interpretable model-agnostic explanations

ELODIE NGOIE MUTOMBO[1], MIKE WA NKONGOLO[2*], and MAHMUT TOKMAK[3]
[1]Department of Computer Science, University of Pretoria, Gauteng, South Africa
[2]Department of Informatics, University of Pretoria, Gauteng, South Africa
[3]Department of Management Information Systems, Burdur Mehmet Akif Ersoy University, Turkey

***Correspondence:** mike.wankongolo@up.ac.za
ORCIDs:
ELODIE NGOIE MUTOMBO: https://orcid.org/0009-0004-5876-7682
MIKE WA NKONGOLO: https://orcid.org/0000-0003-0938-113X
MAHMUT TOKMAK: https://orcid.org/0000-0003-0632-4308

Abstract: Cryptojacking, the unauthorised use of computing resources to mine cryptocurrency, has emerged as a critical threat in today's digital landscape. These attacks not only compromise system integrity but also result in increased costs, reduced hardware lifespan, and heightened network security risks. Early and accurate detection is essential to mitigate the adverse effects of cryptojacking. This study focuses on developing a semi-supervised machine learning (ML) approach that leverages an autoencoder for feature extraction and a random forest (RF) model for classification. The objective is to enhance cryptojacking detection while maintaining a balance between accuracy and interpretability. The proposed methodology is further enhanced with explainable artificial intelligence (XAI) techniques such as local interpretable model-agnostic explanations (LIME) to offer insights into model predictions. Results from datasets such as UGRansome and BitcoinHeist indicate that the semi-supervised approach achieves accuracy rates ranging from 70% to 99%. The study demonstrates that the proposed model provides an efficient, interpretable, and scalable solution for real-time cryptojacking detection across various scenarios.

Key words: Cryptojacking detection, semi-supervised learning, explainable AI (XAI), malware analysis, blockchain security

1. Introduction

Cryptojacking attacks have swiftly risen to prominence as a significant cybersecurity threat. These attacks exploit computational power without consent, aiming to generate profits for attackers by secretly mining cryptocurrencies [5]. Cryptojackers employ various methods, such as malicious websites, malware distribution, compromised servers, and file-less techniques, to execute cryptocurrency mining scripts without the knowledge or consent of victims. Cryptojacking malware manifests in both web-based and host-based variants [5, 24]. In-browser cryptojacking, where mining scripts run in visitors' web browsers, has become increasingly prevalent [24]. These attacks can lead to decreased system performance, increased energy consumption, and financial losses for victims [5, 24]. The emergence of cryptojacking as a service (CaaS) has further facilitated easier access to cryptojacking tools and infrastructure for cybercriminals [5], contributing to the proliferation of these attacks and posing significant risks to individuals and organisations. In addition, the rise in cryptojacking

*Correspondence: mike.wankongolo@up.ac.za

malware is largely fueled by the growing interest in cryptocurrencies, particularly Bitcoin (BTC), which makes tracking and understanding malicious activities challenging. Moreover, the limited research in the field highlights a noticeable lack of novel datasets to address the issue of cryptojacking detection. As a result, proactive measures, such as threat intelligence using machine learning (ML), datasets construction, and explainable artificial intelligence (XAI), are essential for effectively mitigating the risks associated with cryptocurrency mining attacks [5, 24]. This research seeks to overcome these limitations by developing a highly efficient detection system for cryptojacking malware, ensuring enhanced protection for users' devices and resources. The study makes key contributions to the detection of web-based cryptojacking malware by addressing the limitations and gaps present in existing research. These contributions collectively advance the field of cryptojacking detection, offering novel insights and innovative features to improve the effectiveness of malware detection and analysis. Generally, the lifecycle of web-based cryptojacking begins with the attacker registering with a service provider to obtain a unique application programming interface (API) key [5]. This key is embedded into a malicious script, which is then injected into a public website.

When unsuspecting users visit the compromised site, the script executes in the background, exploiting the users' device resources to mine cryptocurrency [5]. Since the script is tied to the attacker's API key, the mined cryptocurrency is automatically credited to them. This lifecycle highlights the stealthy and rapid proliferation of web-based cryptojacking malware, posing a significant threat to users' privacy and device security. Different ML algorithms are employed to classify and detect crytojacking attacks. Unfortunately, the interpretability of results poses challenges, prompting research into XAI techniques to make ML models more transparent, interpretable, and understandable, thereby enhancing reliability and effectiveness in recognising and mitigating cyberattacks [1]. While there has been extensive research on combating the spread of cryptojacking [12, 24] there has been relatively little focus on early detection methods.

Traditional signature-based models are ineffective against new attacks, and studies have shown that over 68 % of infections spread extensively within breached networks before being detected [3]. And detecting crypto ransomware within the first 30 minutes of infiltration could potentially save up to 95 % of costs [3]. Furthermore, recent advancements suggest that monitoring system APIs, tracking registries, analysing file activities, and examining network traffic metadata hold promise for identifying cryptomining attacks. Nevertheless, current tools are primarily tested in simulated environments, and their effectiveness in real-world scenarios still needs to be verified. Also, challenges persist in collecting sufficient data for training and obtaining accurate ground truths with high precision. As such, this study uses the UGRansome [3] and BitcoinHeist [17] datasets to promptly detect behavioural anomalies that may indicate cryptojacking to enable swift response and damage control. The methodology utilises ML techniques, employing an autoencoder for feature selection and random forest (RF) for classification. Following this, the study applies local interpretable model-agnostic explanations (LIME), an XAI technique, to effectively predict and interpret cryptojacking features. Based on the provided context, the main research question can be formulated as follows: How can ML techniques, specifically the combination of autoencoders for feature selection and RF models for classification, be enhanced using XAI methods like LIME to effectively predict and interpret cryptojacking activities?

The research evaluates the performance of autoencoder and RF models for early cryptojacking detection, focusing on metrics such as accuracy, precision, recall, ROC, and F1 score to identify the most effective algorithm in diverse cryptomining scenarios. It further enhances the interpretability of ML results by integrating LIME, which provides deeper insights into feature importance and model predictions, uncovering the mechanisms behind cryptojacking recognition. The research hypothesises that combining autoencoder and RF yields

high detection performance, LIME improves explainability, and experimental datasets enable comprehensive cryptojacking classification. Various studies have highlighted deficiencies in existing methods to detect cryptojacking [14]. For instance, DeCrypto Pro, a framework that selects the optimal ML algorithm based on system resources is developed in [20]. The authors classified applications as benign or malicious using performance counters like CPU usage, with a dataset containing both benign and malicious software. The results showed F1-scores of 95.5 % for long short-term memory (LSTM), 97.62 % for RF, and 89.99 % for K-nearest neighbor (KNN). The effectiveness of ML algorithms for detecting cryptojacking malware is experimented in [8]. Their dataset comprised 220 benign samples and 1,500 cryptojacking Windows portable executables (PE) from VirusTotal. The study employed both static and dynamic analysis. The former uses models such as LSTM, attention-based LSTM, and convolutional neural network (CNN), achieving an accuracy of 95 % [8].

The latter involves running the malware in a sandbox to capture system call sequences, resulting in a 99 % success rate. CryingJackpot, an intrusion detection system (IDS) for cryptojacking recognition is presented in [13]. This IDS utilised unsupervised learning to analyse system events and network flows by employing clustering algorithms such as K-means and DBSCAN. When evaluated on the CSECIC-IDS2018 dataset, it achieved an F1-score of 82 %, with subsequent evaluations reaching an F1-score of 97 %. MinerGuard, an artificial neural network (ANN) that monitors CPU usage to detect cryptojacking malware is demonstrated in [27]. Their dataset consisted of 850 websites, and MinerGuard achieved an accuracy of 99 %, including the detection of zero-day attacks. However, it relies on the development version of Google Chrome processes API [27]. A hybrid approach used in [16] combines block list comparisons, static signatures, and dynamic analysis. Tested on 1,000 samples, with 30 identified as malicious, their method achieved an accuracy of 99.6 %. Different ML models were employed to detect zero-days attacks by analysing traffic meta-data [4]. The challenges of accurately selecting impactful features and the high rate of false alarms in ML are highlighted in [4], emphasising the need for explainability techniques. XAI methods like local interpretable model-agnostic explanations (LIME) offer valuable insights into key features that can potentially enhance the interpretability of ML results for detecting zero-days attacks, such as cryptojacking [26]. Building on this, the study aims to improve cryptojacking detection by utilising the UGRansome and BitcoinHeist datasets, which contain over 80,000 samples. Multiple detection models, including autoencoder, RF, and XAI, are evaluated based on metrics such as accuracy, precision, recall, ROC, computational speed, and F1 score to identify the best-performing model. Furthermore, these datasets are employed to apply various cryptojacking analysis approaches.

The goal is to demonstrate the potential of these datasets in facilitating effective cryptojacking detection. Hence, the research proposes a hybrid/semi-supervised model to improve ML interpretability, offering a more robust approach for detecting, classifying, predicting, and understanding cryptojacking. The limitations of the existing literature on cryptojacking detection can be attributed to several key factors. First, many of the current datasets are large, unstructured, and noisy, highlighting the need for the development of novel, well-curated cryptojacking datasets. Second, a lack of interpretability in many approaches hampers the ability to effectively identify malicious transactions, addresses, and activities specific to cryptocurrency mining. Finally, there is a pressing need for researchers to explore advanced ML techniques for dimensionality reduction and feature extraction to enhance detection capabilities and model performance [10, 14]. For example, while ML has been employed to classify zero-day exploits using the UGRansome dataset [3, 25], specific characteristics and interpretability of cryptojacking have yet to be noticed. In addition, certain methods, including recurrent neural networks (RNNs) such as LSTM, gated recurrent unit (GRU), and simple RNN, heavily rely on legacy datasets like UNSW-NB15 and NSL-KDD [15].

This reliance may restrict their applicability to novel malware types. Although current detection methods show potential, their success may differ depending on the type of attacks, system setup, and operating conditions. Even with improved detection methods, novel malware keeps changing, posing fresh hurdles that demand ongoing research and innovation. The manuscript is structured as follows: Section 2 introduces the proposed methodology by illustrating the experimental datasets. Section 3 presents the semi-supervised methods, including the selected ML algorithms, and evaluation metrics. Section 4 discusses the results using LIME. Lastly, Section 5 concludes.

2. Methodology

Experimental steps delineating the research methodology for implementing the semi-supervised model consists of pre-processing experimental datasets, refining features, choosing appropriate ML models, and carrying out experimental tests to validate the selected approach (Figure 1). These steps have been stratified into (i) feature engineering, (ii) data encoding, (iii) feature extraction, (iv) ML, and (v) LIME (Figure 1). This section outlines the key characteristics and attributes of the experimental datasets.

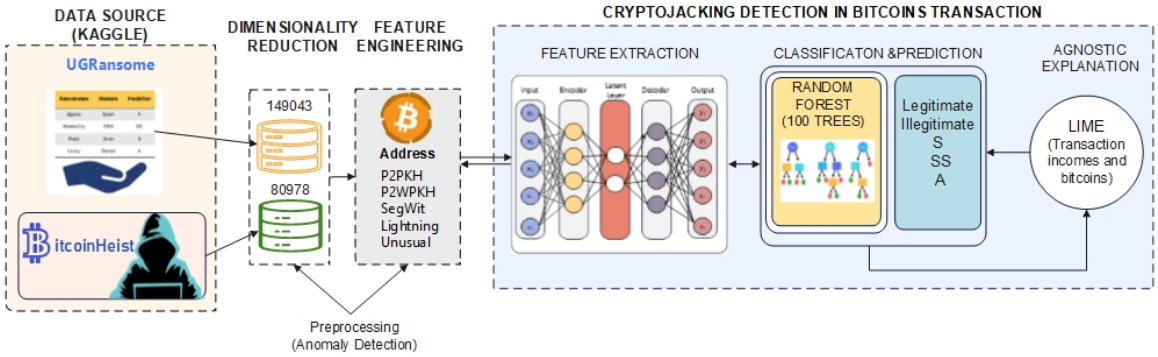


Figure 1: The proposed semi-supervised ML Model.

2.1. The UGRansome dataset

The UGRansome dataset is a key resource for identifying zero-day exploits and ransomware [22, 28]. It is a new cybersecurity dataset created in 2021. The dataset is unique in its inclusion of previously undocumented zero-day attacks, covering a wide range of notorious ransomware families such as WannaCry, Locky, and Crypto Locker, along with advanced persistent threats (APT) such as JigSaw, SamSam, and TowerWeb [3, 7, 25]. The original sample contains 207,533 data points, each characterised by 14 distinct attributes. The dataset is selected for its large sample size, which supports effective training and testing of ML models [23]. The UGRansome dataset is well-suited for cryptojacking detection because it includes previously undocumented zero-day attacks, covering a variety of APTs [7, 21]. This makes it highly relevant for identifying emerging attack patterns, including cryptojacking, which is often coupled with ransomware tactics. Compared to other datasets, UGRansome is also designed for signature-based detection or dynamic analysis [3, 25]. The publicly available dataset uses a CSV format with a total file size of 10.0 MB, including 17 ransomware families[1]. The non-redundant version of the dataset contains a total of 149,043 instances.

[1] https://www.kaggle.com/datasets/nkongolo/ugransome-dataset/data

2.2. The BitcoinHeist dataset

The BitcoinHeist dataset was created by collecting real-world bitcoin transaction data from multiple sources, including public blockchain records and known cryptocurrency thefts or scams [2]. The dataset includes transactions downloaded and parsed from January 2009 to December 2018 using a 24-hour interval[2]. Daily transactions were extracted on the network to form the BTC graph [2]. Network edges that transferred less than 0.3 BTC were filtered out [2]. In 24 ransomware families, at least one address appears in more than one 24-hour time window. Crypto Locker has 13 addresses that appear more than 100 times each [2]. The Crypto Locker address 1LXrSb67EaH1 appears for a maximum of 420 times [2]. Four addresses have conflicting ransomware labels between Montreal and Padua data sources [2]. JigSaw APTs have two and one pay-to-script-hash (P2SH) addresses that start with 3 in the Montreal and Padua data [2]. All other addresses are ordinary addresses that start with 1. This dataset has 2,916,697 features with ten attributes (Table 1). It includes different ransomware, covering many notorious families. The BitcoinHeist dataset includes features like income, neighbors, weight, and loop, which can help model the transactional behavior of cryptocurrency wallets involved in illicit activities, including cryptojacking (Table 1). It covers ransomware families like Cryptowall, Locky, and WannaCry, which are closely related to cryptojacking since both exploit computational resources for financial gain. Identifying ransomware wallet patterns may indirectly assist in detecting cryptojacking transactions [2].

Table 1: The BitcoinHeist dataset's description

Attributes in the BitcoinHeist dataset				
Column	**Description**	**Type**	**Definition**	**Example**
1	Address	Qualitative	Bitcoin address associated with transactions.	1LJtBHHV5S
2	Year	Quantitative	Year when transactions occurred.	2013
3	Day	Quantitative	Day of the year when the transaction occurred.	120
4	Length	Quantitative	Length of the transaction (number of characters).	350
5	Weight	Quantitative	Weight of transactions.	0.0025
6	Count	Quantitative	Number of transactions involving the same address.	10
7	Looped	Quantitative	Number of times the transaction is repeated.	1
8	Neighbors	Quantitative	Number of distinct addresses.	5
9	Income	Quantitative	Total income received in the transaction.	2.5 BTC
10	Label	Qualitative	Target transaction variable.	Legitimate or illegitimate

2.3. Experimental datasets comparison

The UGRansome and BitcoinHeist datasets are available on Kaggle [2]. They contain ransomware data, making them well-suited for training and testing ML models for cryptojacking recognition. The UGRansome dataset provides a comprehensive set of features, such as timestamps, network protocols, ransomware families, and monetary values in BTC. These attributes allow for the detailed analysis of ransomware behaviors across various

[2] https://www.kaggle.com/datasets/sapere0/bitcoinheist-ransomware-dataset

attacks, including botnets, DoS, and port scanning. Similarly, the BitcoinHeist dataset includes attributes like BTC addresses (e.g., P2SH), transaction years, income, and loops indicating legitimate or illegitimate transactions. Therefore, the rich feature sets in both datasets will enable robust feature extraction, classification, and prediction tasks, for effectively engineering and identifying cryptojacking activities.

2.4. Feature engineering

During the feature engineering phase, the study refined the UGRansome dataset by removing three irrelevant attributes, namely flags, ports, and USD. This refinement process resulted in creating the UGRansome2024 dataset, a streamlined version aimed at reducing dimensionality and enhancing computational efficiency. Specific BTC addresses within both datasets were renamed to align with their respective transaction types. For instance, addresses starting with the number 1 were re-labelled as pay-to-public-key-hash (P2PKH), while those beginning with 3 were renamed to pay-to-script-hash (P2SH) (Figure 1). Furthermore, addresses containing the character *bc*1 were renamed to pay-to-witness-public-key-hash (P2WPKH) to reflect the SegWit upgrade in the blockchain network [2]. The BitcoinHeist dataset underwent substantial optimisation, resulting in a streamlined dataset with 80,978 rows and 10 columns (Figure 1). After a thorough examination, both datasets were free of missing values and duplicate entries. The study uses a Python label encoder to encode categorical features in both datasets.

3. Method

An autoencoder is used as a type of ANN for unsupervised and deep learning (DL). It consists of two main components: an encoder that compresses the input data into a lower-dimensional representation, and a decoder that reconstructs the extracted input from this representation. Let $\mathbf{X}$ be the input data, f_{enc} be the encoding function, and f_{dec} be the decoding function. The goal of training an autoencoder is to minimise the reconstruction error, typically measured using a loss function such as mean squared error (MSE). In the autoencoding process, θ represents the parameters of both the encoder and decoder, and n is the number of training examples. The encoder maps the input data $\mathbf{X}$ to a lower-dimensional representation $\mathbf{Z} = f_{\text{enc}}(\mathbf{X})$. This process can be mathematically represented as shown in Equation 1.

$$\mathbf{Z} = \sigma(\mathbf{W}\mathbf{X} + \mathbf{b}). \tag{1}$$

Where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, and σ is the activation function. The decoder then reconstructs the original input $\mathbf{X}' = f_{\text{dec}}(\mathbf{Z})$ using the encoded representation $\mathbf{Z}$. This is achieved through another ANN layer, which maps $\mathbf{Z}$ back to $\mathbf{X}'$. The autoencoder learns a compact representation of the input data by minimising the reconstruction error, capturing its essential features in the lower-dimensional space. In turn, RF serves as an ensemble learning method employed for classification tasks [15]. It builds multiple decision trees by randomly selecting subsets of the training data and features, which helps reduce overfitting and improve generalisation performance [15]. The final prediction is determined by a majority vote among the individual trees [15].

3.1. Anomaly detection techniques

The isolation forest (IF) algorithm is used to detect anomalies by isolating observations through random splits [19]. The anomaly score $S(x)$ for a given observation x is defined in Equation 2. For a given dataset X with n observations and n features, the IF is described as

$$S(x) = \frac{E(h(x))}{c(n)}, \tag{2}$$

where $E(h(x))$ is the average path length of x, and $c(n)$ is the average of unsuccessful searches depicted in Equation 3.

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \tag{3}$$

with $H(i)$ being the i-th harmonic number illustrated in Equation 4.

$$H(i) = \sum_{k=1}^{i} \frac{1}{k}. \tag{4}$$

If $S(x)$ is close to 1, then x is considered an outlier. Otherwise, it is considered normal. The autoencoder detects anomalies based on the reconstruction error $L(x, \hat{x})$, computed as the MSE between the original input x and its reconstruction $\hat{x}$ (Equation 5).

$$L(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2. \tag{5}$$

A threshold ϵ is set such that if $L(x, \hat{x}) > \epsilon$, the observation x is flagged as an anomaly. Pre-processing these anomalies is crucial for enhancing the performance and reliability of the semi-supervised model. Anomalies, such as outliers and missing values, could significantly skew results and lead to inaccurate predictions. The study addressed these issues by pre-processing anomaly data, which improves accuracy and enhances the robustness of the proposed model, making it less sensitive to noise and variations in the data. Furthermore, clean and well-preprocessed data reduced training time and the computational resources required, making the entire modeling process more efficient.

3.2. Local interpretable model-agnostic explanations (LIME)

LIME is utilised to explain individual predictions by perturbing the input data and observing changes in the output [11]. The formula for computing the value (or weight) of a feature involves the following steps:

1. **Generate perturbed samples**
   Perturbed samples are generated around the original input sample x, denoted as $\tilde{x}$. The equation for generating the perturbed samples is given by:

   $$\tilde{x}_i = x + \epsilon_i \quad \text{for each perturbed sample } \tilde{x}_i, \tag{6}$$

   where ϵ_i represents small perturbations added to the original sample x.

2. **Model predictions**
   Next, predictions are made by the model on the perturbed samples $\tilde{x}_i$. The prediction function is defined in Equation 7.

   $$\hat{y}_i = f(\tilde{x}_i), \tag{7}$$

where f is the black-box model (in this case, the RF), and $\hat{y}_i$ is the prediction for the perturbed sample $\tilde{x}_i$.

3. **Weighting samples**
The generated samples are weighted based on their similarity to the original instance x. The weighting function is defined in Equation 8.

$$w_i = \exp\left(-\frac{D(x,\tilde{x}_i)^2}{2\sigma^2}\right), \tag{8}$$

where $D(x,\tilde{x}_i)$ is an Euclidean distance between x and $\tilde{x}_i$, and σ is a kernel width parameter [11].

4. **Linear model fitting**
Finally, a linear model is fit to the weighted samples to interpret the predictions. The linear model is defined in Equation 9.

$$g(z) = \sum_j \theta_j z_j, \tag{9}$$

where $g(z)$ is the linear model, z_j represents the features, and θ_j are the coefficients that represent the importance of each feature [11]. This study implements LIME as a post-hoc method to generate explanations after the autoencoder and RF models have been trained [6], (Figure 1). The aim is to enhance the interpretability of cryptojacking detection in ML-based XAI systems.

3.3. Data pre-processing and feature selection

The datasets were pre-processed, reducing UGRansome to 149,043 and BitcoinHeist to 80,978 observations (Figure 1). Addresses were labeled as legitimate, illegitimate, signature (S), synthetic signature (SS), and anomaly (A) (Figure 1). And a 3-layer autoencoder selected key features from these addresses, with RF classifying and predicting cryptojacking transactions (Figure 1). Lastly, LIME highlighted various relevant features influencing cryptojacking attacks (Figure 1).

3.4. Parameters fine-tuning

Table 2 outlines the parameters of the proposed model. For the autoencoder, the parameters include the Adam optimiser, MSE loss function, rectified linear unit (Relu) as the activation function for hidden layers, and Sigmoid activation function for the output layer.

Table 2: Parameters of the proposed semi-supervised model

	Parameters	**UGRansome (sec)**	**BitcoinHeist (sec)**
Autoencoder	Adam, ReLU, MSE & Sigmoid	3.5790	4.763
RF	Gini & Entropy	0.3281	2.438
LIME	RF & Logit	1.0614	2.143

These parameters are used for training the autoencoder. The Adam optimiser is an adaptive learning rate optimisation algorithm, MSE is a common loss function for regression tasks, and Relu and Sigmoid are activation functions used in the ANN layers [9]. In turn, Gini and entropy measured the quality of an RF split, with the number of trees in the forest set to 100 (see Table 2). The LIME model is set to RF with a Logit link

function to generate local explanations for RF predictions (Table 2). The link function specifies transformations applied to the output of the interpretable model [11, 26]. The study uses Python libraries such as numpy, Keras, TensorFlow, pandas, matplotlib, seaborn, scikit-learn, and lime to implement the semi-supervised model. The execution times for each ML models were measured as follows: autoencoder at 3.5790 seconds, RF at 0.3281 seconds, and LIME at 1.0614 seconds (Table 2).

3.5. Data split

The study used the train test split function from the Python scikit-learn library to randomly partition the experimental datasets into a training and testing sets while preserving the original class distribution. The training set comprised 80% of the total samples, and the testing set the remaining 20% [15]. Instead of creating a separate validation set, four-fold cross-validation is employed to provide a comprehensive performance evaluation of the models and to mitigate potential biases in the data split.

3.6. Evaluation metrics

The evaluation metrics used to assess the proposed model are detailed in Table 3, with true positive (TP), true negative (TN), false positive (FP), and false negative (FN) defined as per [18]. Table 4 provides a confusion matrix used to evaluate the classification performance in identifying cryptojacking transactions. In this context, TP refers to transactions correctly classified as cryptojacking. FP denotes transactions incorrectly identified as cryptojacking. Conversely, FN represents misclassified cryptojacking transactions. TN encompasses transactions accurately classified as non-cryptojacking.

Table 3: Evaluation metrics and their descriptions

Equation	Description
$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$	Measures the proportion of correctly classified samples.
$\text{Precision} = \frac{TP}{TP+FP}$	Represents the proportion of samples accurately classified as positive.
$\text{Recall} = \frac{TP}{TP+FN}$	Represents the ratio of correctly classified positive samples to the total number of actual positive samples.
$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$	The harmonic mean of precision and recall.
$\text{ROC} = \frac{TP}{TP+FN}, \quad \text{FPR} = \frac{FP}{FP+TN}$	Trade-off between TP rate (TPR/sensitivity) and FP rate/FPR (1-specificity) across thresholds.

Table 4: Confusion matrix

Actual / predicted	Positive (cryptojacking)	Negative (non-cryptojacking)
Positive (cryptojacking)	TP	FP
Negative (non-cryptojacking)	FN	TN

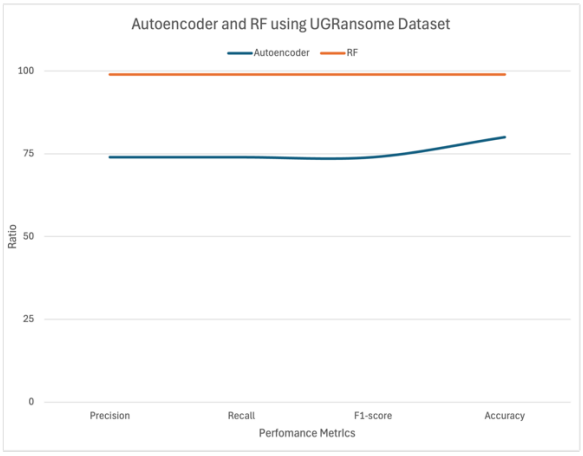
4. Results and discussion

The semi-supervised model underwent evaluation using a four-fold cross-validation approach, and the reported performance is an average across all folds. As highlighted in Table 5 and Figure 2, the findings offer valuable insights into the effectiveness of the autoencoder and RF algorithms in classifying cryptojacking attacks. The RF model demonstrated commendable performance across various metrics showcasing moderate to high accuracy

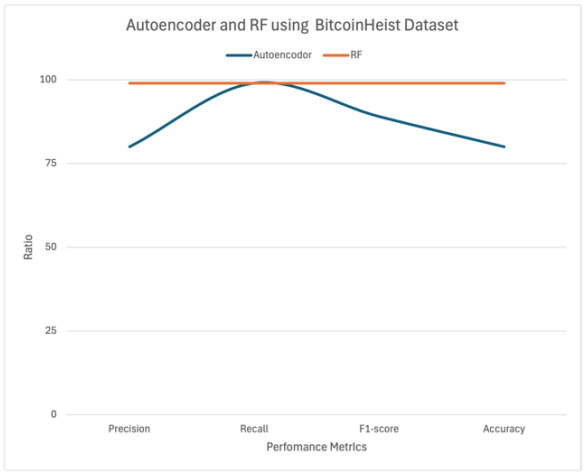
and precision, hovering around 99% (Table 5 and Figure 2). This performance rate indicates its ability to differentiate between different classes of cryptojacking effectively compared to other ML algorithms (Figure 3). The graph in Figure 2 exhibits moderate level of recall, implying that it makes reasonable predictions overall (Figure 3).

Table 5: Performance metrics of the semi-supervised model

Performance using the BitcoinHeist dataset				
Prediction	**Precision**	**Recall**	**F1-score**	**Support**
0	0.80	0.99	0.89	52,447
1	0.61	0.05	0.09	13,553
			Accuracy	80%
Performance using the UGRansome dataset				
Prediction	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	0.99	0.99	12,662
1	0.99	0.99	0.99	19,986
2	1.00	0.99	0.99	12,065
			Accuracy	99%



(a) Evaluation metrics using UGRansome data.



(b) Evaluation metrics using BitcoinHeist data.

Figure 2: Performance metrics for the BitcoinHeist and UGRansome datasets.

In contrast, the graph in Figure 4 illustrates autoencoder displaying lower performance metrics compared to the RF. Figure 5 demonstrates the autoencoder's classification challenges using the BitcoinHeist dataset where target variables are encoded into 1 (legitimate transactions) and 0 (illegitimate transactions). The RF experiments demonstrated performance levels on par with or surpassing those of the autoencoder model in most trials.

4.1. XAI using LIME

Table 6 shows the cryptojacking features provided by LIME for transaction recognition. These features are strongly related to cryptojacking because they help identify suspicious behaviors and patterns commonly associated with illicit cryptocurrency mining.

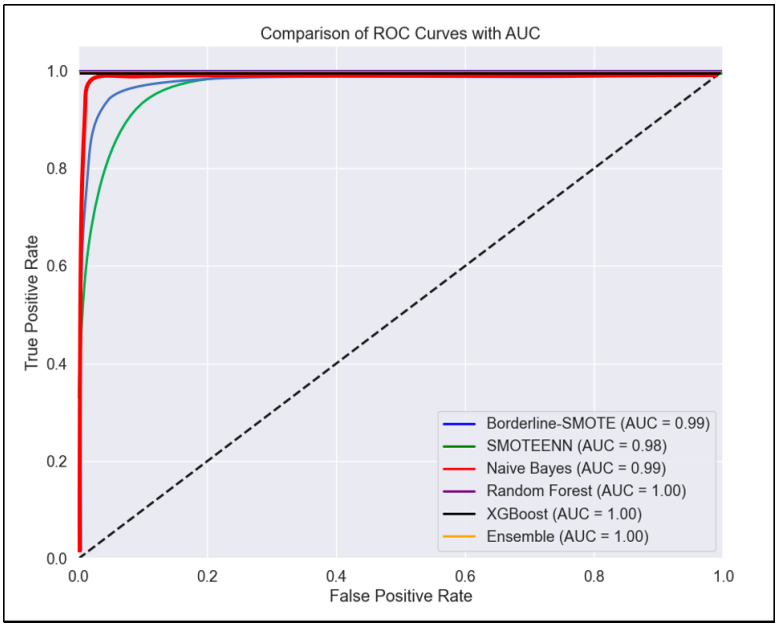
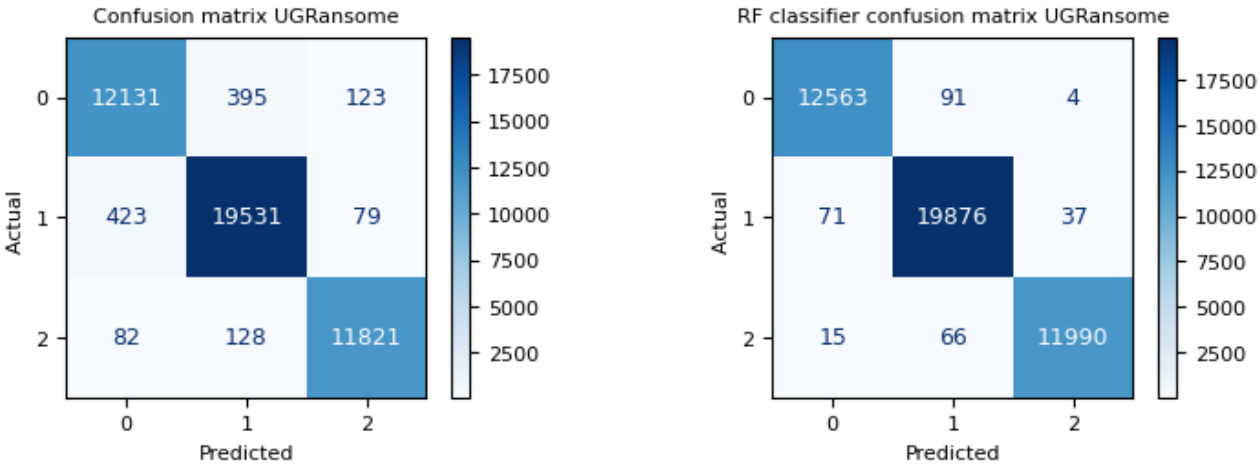


Figure 3: Comparison between model AUC.

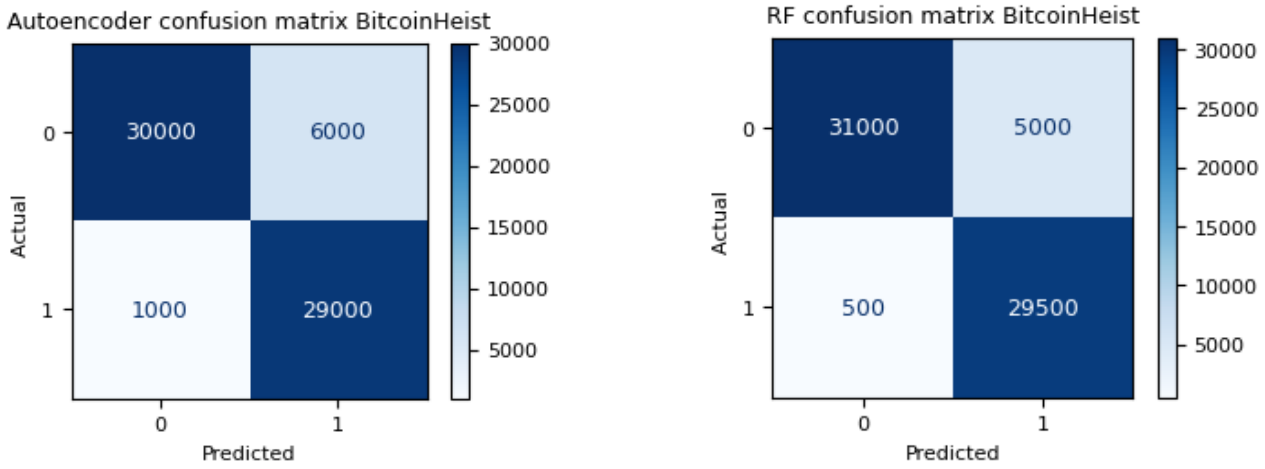


(a) Autoencoder confusion matrix.

(b) RF confusion matrix.

Figure 4: Confusion matrices for the autoencoder and RF models.

The feature values of address and income indicate compromised wallets and unexpected financial gains, which are characteristic of cryptojacking operations (Table 6). Similarly, weight and loop values highlight unusual computational loads and repetitive transactions, often used to conceal malicious activities (Table 6). The family and threats values signify the malware type involved, with specific families linked to cryptojacking. Features such as P2SH, P2PKH, and SegWit (e.g., Bech, P2WPKH) represent transaction formats often exploited for cryptojacking activities, allowing attackers to blend their transactions within legitimate traffic, (Table 6). Timestamp values help detect abnormal mining patterns over time, while IP addresses can provide insights into the origin of these malicious activities, such as botnets.



(a) Autoencoder confusion matrix.

(b) RF confusion matrix.

Figure 5: Misclassification results of the BitcoinHeist dataset.

These features, when analysed in combination, offer a comprehensive view of cryptojacking detection and classification (Table 6). In Figure 6, incomes recorded in BTC contributed to illegitimate transaction prediction by approximately 24%. Figure 7 shows that 100% of signature (S) attack families with seed addresses contribute to this prediction. Figure 8 illustrates BTC features that contribute to the prediction of seed addresses in cryptojacking detection. The figure emphasizes significant features such as expended addresses and network protocol, which strongly influence the model's decisions when identifying SegWit transactions. The quantitative values of these features provide insight into their contributions, enhancing the model's ability to detect signature (S) attacks with high accuracy. This capability is essential for enabling timely interventions and mitigating the damage caused by malicious cryptomining activities.

Table 6: LIME's features values for cryptojacking recognition

Feature	**Value 1**	**Value 2**	**Value 3**	**Value 4**	**Prediction**	**Probability**
Address	1.00	-	-	-	Legitimate	76%
Income	300.00	-	-	-	Illegitimate	24%
Weight	0.20	-	-	-	Legitimate	25%
Loop	15.00	-	-	-	Illegitimate	24%
Family	1.40	1.40	-	1.40	Signature (S)	100%
Threats	0.20	0.20	-	0.20	Signature (S)	30%
BTC	5.10	3.50	-	-	Signature (S)	5%
Seed address	3.50	-	-	-	Signature (S)	100%
P2SH	1.40	-	-	-	SegWit	100%
Bech	0.20	-	-	-	SegWit	30%
P2PKH	5.10	-	-	-	SegWit	100%
P2WPKH	3.50	-	-	-	SegWit	100%
Timestamp	5.10	-	5.10	-	Signature (S)	100%
IP	3.50	-	-	-	Signature (S)	3%

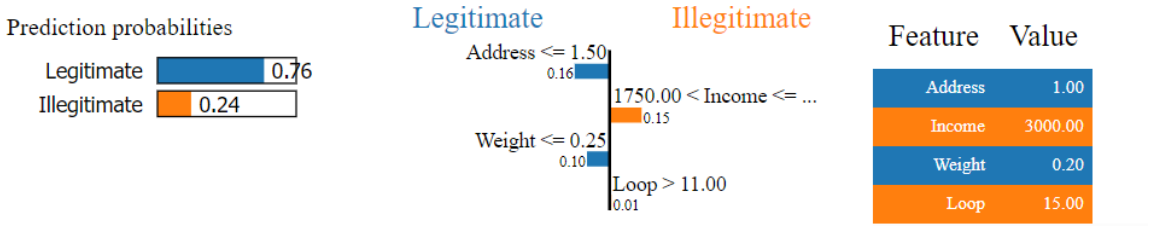


Figure 6: LIME results using the BitcoinHeist dataset.

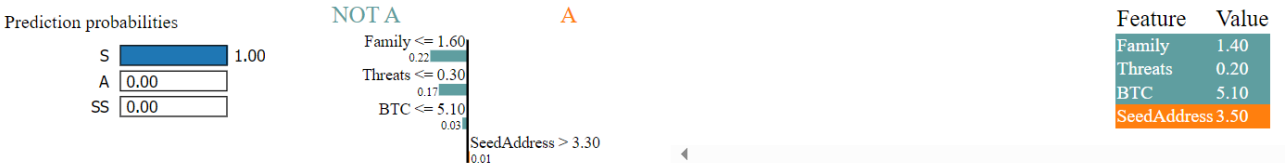


Figure 7: Prediction results using the BitcoinHeist dataset.

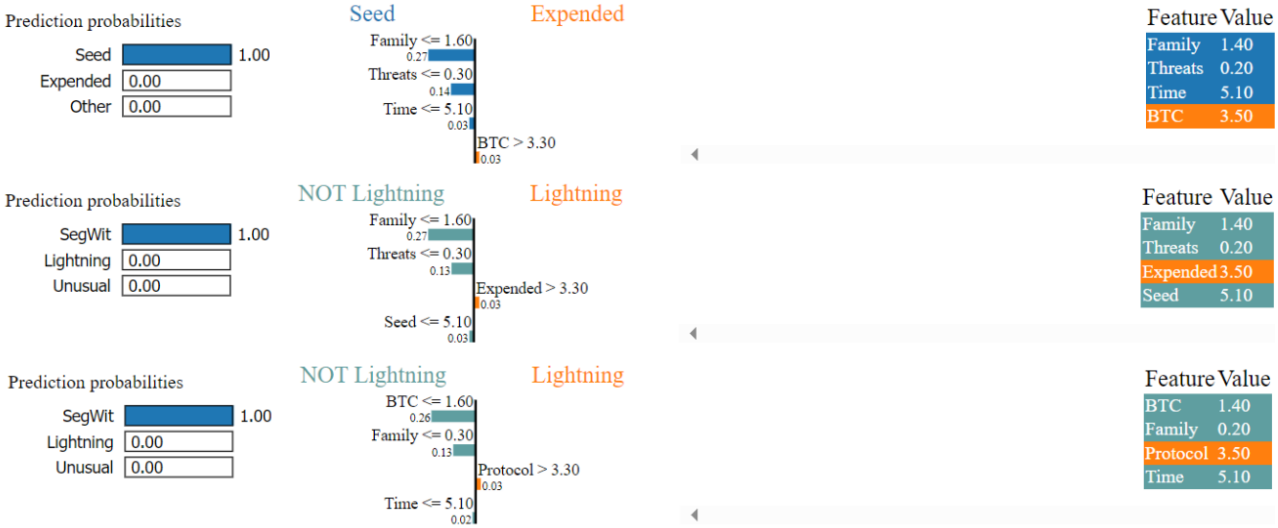


Figure 8: Transaction prediction using LIME.

4.2. Discussion

This study distinguishes itself from previous research in several key aspects. While Mani et al. [20] and Darabian et al. [8] focused on the application of ML classifiers such as LSTM, RF, and K-NN for web-based and host-based cryptojacking detection, our research innovatively combines autoencoders, RF, and LIME, addressing both high detection accuracy and model interpretability. This contrasts with studies like Gomes et al. [13], who employed unsupervised learning techniques like K-means and DBSCAN for cryptojacking detection, but did not integrate interpretability into the process. Wu et al. [27] used ANN models to monitor CPU usage for cryptojacking detection, with a focus on performance in web-based attacks, but did not include interpretability or a semi-supervised learning approach [27]. Khan Abbasi et al. [16] proposed a hybrid method with blocklist comparisons and dynamic analysis to detect host-based cryptojacking, yet their focus is more on

improving performance through a combination of techniques rather than interpretability [16]. In comparison, the proposed study emphasises not only detection accuracy and anomaly detection through autoencoders, but also offers insights into feature contributions using LIME, which is particularly important for understanding the underlying factors driving cryptojacking activity, such as BTC incomes and signature (S) attacks through seed addresses. Thus, while prior works focus primarily on detection accuracy, our approach enhances transparency and adaptability with a semi-supervised framework, better suited for real-world applications with limited labeled datasets.

5. Conclusion

This study aimed to investigate how advanced ML techniques, specifically autoencoder, RF, and LIME, can be effectively combined to detect and understand cryptojacking attacks using the UGRansome and BitcoinHeist datasets. The research question is addressed through a series of evaluations and analyses. The findings demonstrate that the RF model achieves commendable performance across various metrics, showcasing high accuracy and precision. The model exhibited moderate recall, indicating reasonable overall predictions, whereas the autoencoder displayed lower performance metrics. Integrating LIME provides more profound insights into feature values and model predictions, enhancing the interpretability of the ML results. Specifically, BTC incomes with seed addresses contributed to illegitimate transactions. The study supports the hypothesis that the RF model, combined with LIME, offers a robust and interpretable approach to cryptojacking recognition, outperforming the autoencoder in this context. The blockchain specific findings highlight the model's utility in analysing blockchain transactions using experimental datasets. These insights can lead to timely interventions and mitigation of the damage caused by malicious cryptomining activities. Future research should explore the scalability of these models in more diverse and extensive datasets to ensure comprehensive and practical cryptojacking recognition.

References

[1] Alsadig Ahmed. Enhancing cybersecurity in financial services using single value neutrosophic fuzzy soft expert set. *International Journal of Neutrosophic Science (IJNS)*, 24(2), 2024.

[2] Cuneyt Gurcan Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain. *arXiv preprint arXiv:1906.07852*, 2019.

[3] Asma A Alhashmi, Abdulbasit A Darem, Ahmed B Alshammari, Laith A Darem, Huda K Sheatah, and Rachid Effghi. Ransomware early detection techniques. *Engineering, Technology & Applied Science Research*, 14(3):14497–14503, 2024.

[4] Omar M. K. Alhawi, James Baldwin, and Ali Dehghantanha. *Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection*, pages 93–106. Springer International Publishing, Cham, 2018. ISBN 978-3-319-73951-9. . URL https://doi.org/10.1007/978-3-319-73951-9_5.

[5] Hadeel A. Almurshid, Iman Almomani, M. A. Khalifa, and Walid El-Shafai. A holistic intelligent cryptojacking malware detection system. *IEEE Access*, 12:161417–161439, 2024. .

[6] Marilyn Bello, Rosalís Amador, María-Matilde García, Javier Del Ser, Pablo Mesejo, and Óscar Cordón. The level of strength of an explanation: A quantitative evaluation technique for post-hoc xai methods. *Pattern Recognition*, page 111221, 2024.

[7] I. Chaudhary and S. Adhikari. Ransomware detection using machine learning techniques. *Researcher CAB: A Journal for Research and Development*, 3(1):96–114, 2024. . URL https://doi.org/10.3126/rcab.v3i1.68424.

[8] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza M Parizi, and Kim-Kwang Raymond Choo. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, 18:293–303, 2020.

[9] L Dhanya and R Chitra. A novel autoencoder based feature independent ga optimised xgboost classifier for iomt malware detection. *Expert Systems with Applications*, 237:121618, 2024.

[10] Omar Dib, Zhenghan Nan, and Jinkua Liu. Machine learning-based ransomware classification of bitcoin transactions. *Journal of King Saud University-Computer and Information Sciences*, 36(1):101925, 2024.

[11] Diogo Gaspar, Paulo Silva, and Catarina Silva. Explainable ai for intrusion detection systems: Lime and shap applicability on multi-layer perceptron. *IEEE Access*, 12:30164–30175, 2024. .

[12] Hadi Gharavi, Jorge Granjal, and Edmundo Monteiro. Post-quantum blockchain security for the internet of things: Survey and research directions. *IEEE Communications Surveys Tutorials*, 26(3):1748–1774, 2024. .

[13] Gilberto Gomes, Luis Dias, and Miguel Correia. Cryingjackpot: Network flows and performance counters against cryptojacking. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2020.

[14] Laith M Kadhum, Ahmad Firdaus, Syifak Izhar Hisham, Waheed Mushtaq, and Mohd Faizal Ab Razak. Features, analysis techniques, and detection methods of cryptojacking malware: A survey. *JOIV: International Journal on Informatics Visualization*, 8(2):891–896, 2024.

[15] Sydney Mambwe Kasongo. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Computer Communications*, 199:113–125, 2023.

[16] Muhammad Haris Khan Abbasi, Subhan Ullah, Tahir Ahmad, and Attaullah Buriro. A real-time hybrid approach to combat in-browser cryptojacking malware. *Applied Sciences*, 13(4):2039, 2023.

[17] Amit Kumar, Neha Sharma, Rahul Chauhan, Kireet Joshi, Retinderdeep Singh, and Meet Kumari. Detecting ransomware in bitcoin network: Experimental insights. In *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*, pages 1349–1354, 2024. .

[18] Chao Liu, Boxi Chen, Wei Shao, Chris Zhang, Kelvin K. L. Wong, and Yi Zhang. Unraveling attacks to machine-learning-based iot systems: A survey and the open libraries behind them. *IEEE Internet of Things Journal*, 11(11): 19232–19255, 2024. .

[19] Tao Liu, Zhen Zhou, and Lijun Yang. Layered isolation forest: A multi-level subspace algorithm for improving isolation forest. *Neurocomputing*, 581:127525, 2024.

[20] Ganapathy Mani, Vikram Pasumarti, Bharat Bhargava, Faisal Tariq Vora, James MacDonald, Justin King, and Jason Kobes. Decrypto pro: Deep learning based cryptomining malware detection using performance counters. In *2020 IEEE International conference on autonomic computing and self-organizing systems (ACSOS)*, pages 109–118. IEEE, 2020.

[21] Elodie Ngoie Mutombo and Mike Wa Nkongolo. Blockchain security for ransomware detection. *arXiv preprint arXiv:2407.16862*, 2024.

[22] Steven Jabulani Nhlapo and Mike Nkongolo Wa Nkongolo. Zero-day attack and ransomware detection. *arXiv preprint arXiv:2408.05244*, 2024.

[23] Mike Nkongolo. Ransomware detection dynamics: Insights and implications. *arXiv preprint arXiv:2402.04594*, 2024.

[24] Muhammad Saad and David Mohaisen. Analyzing in-browser cryptojacking. *IEEE Transactions on Dependable and Secure Computing*, 21(6):5448–5460, 2024. .

[25] M. Tokmak. Deep forest approach for zero-day attacks detection. In S. Tasdemir and İ. A. Ozkan, editors, *Innovations and Technologies in Engineering*, pages 45–56. Eğitim Yayinevi, Istanbul, Turkey, 2022.

[26] Ilhan Uysal and Utku Kose. Analysis of network intrusion detection via explainable artificial intelligence: Applications with shap and lime. In *2024 Cyber Awareness and Research Symposium (CARS)*, pages 1–6, 2024. .

[27] Min-Hao Wu, Yen-Jung Lai, Yan-Ling Hwang, Ting-Cheng Chang, and Fu-Hau Hsu. Minerguard: A solution to detect browser-based cryptocurrency mining through machine learning. *Applied Sciences*, 12(19):9838, 2022.

[28] Peizhi Yan, Tala Talaei Khoei, Reena Sajad Hyder, and Reemaa Sajad Hyder. A dual-stage ensemble approach to detect and classify ransomware attacks. In *2024 IEEE 15th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 781–786, 2024. .