IN1166 Detecção de Intrusão A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks

1st Edilson A. Silva Junior
Centro de Informática - UFPE
Recife, Brazil
easj@cin.ufpe.br

***Abstract*—Cryptojacking malware attacks takes profit from mining crypto currency by utilizing the processing power of the victim's devices. The previous focus of such attacks were computionally powerful devices, but in the scenario of IoT devices and smart homes, the use of IoT devices at scale, rather than individually, becomes profitable. The problem at hand is to detect cryptojacking attacks in the context of IoT devices and smart homes. Previous related work investigated in-browser and host-based attacks, but none of the earlier works focused on IoT devices and smart home networks. Further on, they did not investigate different attack configurations or network settings. Stealthy cryptojacking malware aims to stay undetected for a longer time, while agressive ones maximize resource utilization to increase profit. To better evaluate the performance of the detection mechanism, fully and partially compromised network setting are also considered. The base article proposes an accurate and efficient IoT cryptojacking detection mechanism based on network traffic features, which can detect both in-browser and host-based cryptojacking. Several novel test scenarios were designed taking into consideration the attack surface, attack configurations and network settings. A dataset of 6.4M network packets was utilized and an accuracy of up to 99% was achieved with only on-hour training data. Some modifications to the orginal article are proposed, but were not implemented due to the long execution time to reproduce the original experiments. The original authors spent over 550 hours (aproximately 23 days) to run the whole set of experiments.**

I. Introduction

A. Motivation and Justification

Cryptocurrency mining, specifically the unauthorized use of victim resources known as cryptojacking, has evolved from targeting personal computers to exploiting the growing Internet of Things (IoT) landscape. IoT devices are attractive targets due to their proliferation in homes and industries, often lacking robust security configurations compared to traditional servers or PCs. While individual IoT devices are resource-constrained, attackers utilize botnets to aggregate their power for profit. The diversity of vendors and protocols in IoT makes unified defense difficult, justifying the need for a network-based, device-agnostic detection mechanism.

B. Main Contributions

The analyzed article presents the following major contributions:

- **Novel Detection Algorithm:** A lightweight, accurate cryptojacking detection mechanism based on network traffic features, achieving 99% accuracy with only one hour of training data.
- **Comprehensive Evaluation Scenarios:** The design of novel experimental scenarios assessing various attack configurations (profit strategies, victim devices, throttle values) and net-

work settings (fully vs. partially compromised networks).
- **Practical Implementation Solutions:** Introduction of novel techniques to overcome practical issues in implementing cryptojacking malware on closed IoT systems like LG WebOS.
- **Open Data and Code:** Release of the dataset and code to accelerate research.

Original Article Code:
https://github.com/cslfiu/IoTCryptojacking
Reproduction code:
https://github.com/edilsoneasj/cryptojackingReproduction
The codes are Jupyter notebooks, with the output results available inside each respective file.

C. Scope and Problem Definition

The scope covers heterogeneous smart home networks containing IoT devices (e.g., Smart TVs, Raspberry Pis) and general-purpose computers. The core problem is detecting unauthorized mining activities that may be either in-browser or host-based.

D. Existing Solutions and Disadvantages

Existing solutions primarily focus on:

1) **Host-based Features:** utilizing CPU/memory counters or instruction execution times.
2) **Browser-based Features:** analyzing JavaScript compilation or execution behaviors.

Disadvantage: These are unsuitable for IoT because most IoT devices (e.g., Smart TVs) are ”black boxes” that do not allow programming to collect hardware-level or browser-specific features. Furthermore, static blacklisting is ineffective due to dynamic domains and IP changes.

II. Related Work

A. Comparison to State of the Art

The literature is categorized into analyzing the cryptojacking ecosystem and detection methods. While some studies utilize network traffic, they often lack device diversity (focusing only on PCs/Servers) or rely on emulated environments rather than real malicious deployments on IoT hardware.

The article also compares its approach to prior works that focused on either in-browser or host-based cryptojacking. The work distinguishes itself by using a wide range of statistical features extracted from network traffic in a real IoT testbed.

B. Comparison Table

Tables I, II and III compare the proposed solution with related works based on features and deployment scope.

III. Modelo de ameaça (se aplicável)

A. Cryptojacking Attack Model

The work evaluates two primary attack vectors utilized by adversaries to monetize victim resources.

1) 1. In-Browser Cryptojacking: Attackers inject JavaScript/WebAssembly scripts into websites. When a user (or IoT device with a browser) visits, the script mines cryptocurrency (e.g., Monero) and reports to a Service Provider (e.g., Coinhive, Webmine).

2) 2. Host-Based Cryptojacking: Attackers install binary malware on the OS. This communicates with a Command and Control (C&C) server or directly with a Mining Pool to receive tasks (hashing puzzles) and submit results (nonces).

IV. Sistema proposto pelo artigo de referência

A. Detection Mechanism Flow

The proposed solution utilizes a machine learning approach based on network traffic metadata, avoiding the need for deep packet inspection (DPI) or on-device agents.

TABLE I: Comparison of Proposed Solution with Related Work

Feature / Approach	CPU/Memory Based	Browser Features	Previous Network Approaches	Proposed Solution
IoT Compatible	No (Requires Access)	No	Partial	**Yes (Network Only)**
Detection Type	Host-based	In-browser	Host-based	**Both**
Environment	Simulation/PC	PC	Emulation/PC	**Real IoT Hardware**
Traffic Analysis	N/A	N/A	Aggregated Stats	**Time-series Features**
Encryption Resilient	Yes	Yes	Yes	**Yes**

TABLE II

Study / Paper	Primary Features Used	Feasibility for IoT (as per authors)
Tekiner et al. (Base Paper)	**Network Traffic (Time-Series Stats)**	**High (Passive, no device agent needed)**
Studies [17], [18], [20], [22], [23], [26]	CPU Events (HPC), Memory Activity	**Low (Requires root/admin access to hardware counters)**
Studies [16], [19]-[28]	Browser-Specific (JS, Wasm execution)	**Low (Not applicable to host-based or non-browser IoT)**
Studies [20], [22], [23]	Aggregated Network + HPC/Memory	**Low (Relies on non-network features unavailable on IoT)**
Studies [21], [73]	Network Traffic (5-6 simple features)	**Medium (Correct feature type, but limited scope)**

TABLE III

Study / Paper	Device Scope	Attack Type Tested	Attacker Strategy Tested?
Tekiner et al. (Base Paper)	**IoT-Specific (Smart TV, Pi, etc.)**	**Type-Agnostic (In-Browser & Host-Based)**	**Yes (Throttling, etc.)**
[21], [73] (Closest Work)	General (non-IoT)	**Host-Based Only**	No
[5], [16], [19]-[28]	General (non-IoT)	**In-Browser Only**	Some (e.g., throttling)
[19], [73]	General (non-IoT)	**Host-Based Only**	No
[18], [67], [71]	General (non-IoT)	**Type-Agnostic**	No

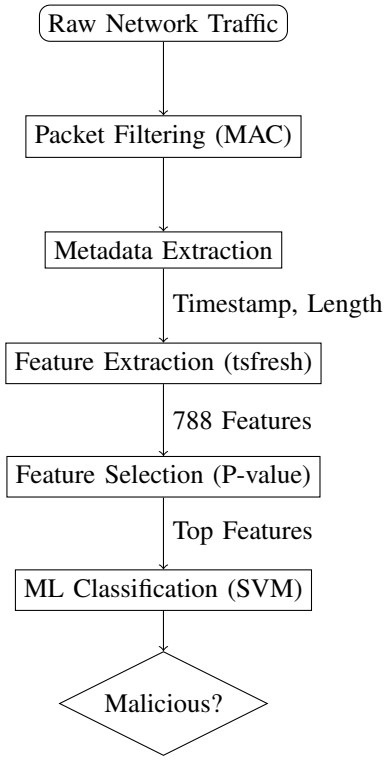

Fig. 1: Flow of the Proposed Detection Mechanism.

B. Feature Extraction and Algorithms

- **Metadata:** The system extracts *timestamp* and *packet length* from bursts of 10 packets.
- **Extraction Tool:** The `tsfresh` library is used to calculate 788 statistical features (e.g., mean, variance, entropy) from the time-series data of packet metadata.
- **Feature Selection:** Statistical significance (P-value) is used to reduce the feature set to approximately 290 relevant features.

C. Analysis of Machine Learning Algorithms

The authors evaluated four classifiers: Logistic Regression (Logreg), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gaussian Naive Bayes (GNB).

Why SVM? Support Vector Machine (SVM) was selected as the optimal classifier because:

1) It achieved the highest metrics across the board (Accuracy: 0.99, F1: 0.98).
2) It is effective in high-dimensional spaces (created by the 788 extracted features).

3) It demonstrated stability; small dataset changes did not drastically alter results, which is crucial for dynamic network environments.

V. Solução proposta pela equipe para melhorar a solução do artigo de referência (CASO FEITA)

The author of the original article show that there is a greater difficulty to detect attacks that use a stealth profit strategy in comparison to aggressive strategies. Also, they note that there is an asymmetry of network data amounts when comparing the different network directions (client-server vs server-client). Also, the tsfresh feature extraction and selection process produced 788 features, that were reduced by the author to 290. This is still a high number of features. To solve that it is being attempted to use dimensionality reduction approches. The 2 chosen approaches are PCA and Autoencoder techniques. Howerver due to the excess amout of time spent to reproduce the article, it was not completed.

VI. Metodologia

A. Dados usados pelo artigo de referência

Data Collection: Available on: (https://github.com/cslfiu/IoTCryptojacking)

A realistic testbed was created using a Raspberry Pi 4 (IoT), LG Smart TV (IoT), Laptop, and Tower Server. Traffic was captured using port mirroring and ARP poisoning.

- **Malicious Data:** Generated by running MinerGate (host-based) and Webmine scripts (in-browser).
- **Benign Data:** Collected from regular user activities (browsing, video, idle).

B. Novo conjunto de dados escolhido

A second public database was chosen by the authors to reproduce the data and it was also reproduced by me. It is a public database called: "Network traffic for machine learning classification" and is available on: https://data.mendeley.com/datasets/5pmnkshffm/1

The data was also grouped on the dataset files of the base article (https://github.com/cslfiu/IoTCryptojacking).

C. Métricas de avaliação

Metrics: The standard classification metrics utilized were Accuracy, Precision, Recall, F1 Score, and Test ROC.

D. Experimental Scenarios

The paper designed 10 distinct scenarios to validate robustness:

1) **Device Variety (Server vs. Desktop vs. IoT):** Testing detection across different hardware capabilities.
2) **Profit Strategies:** Evaluating "Aggressive" (100% CPU), "Robust" (50%), and "Stealthy" (10%) mining throttles.
3) **Cryptojacking Type:** Comparing In-Browser vs. Host-Based malware detection.
4) **Fully Compromised Network:** All devices in the home are mining.
5) **Partially Compromised:** A mix of compromised IoT and Desktop devices.
6) **Single Compromised (IoT):** Only one IoT device is infected (stealthiest network scenario).
7) **IoT Compromised (Multiple):** Two different IoT devices (TV + RPi) are infected.
8) **Imbalanced Dataset:** Testing performance when benign data vastly outweighs malicious data.
9) **Transferability:** Training on one device/malware type and testing on another (e.g., Train on IoT, Test on WebOS).
10) **Non-Default Parameters:** Analyzing the impact of tuning SVM kernel and Gamma parameters.

VII. Resultados e discussões

A. Performance Overview

The system demonstrated high efficacy across the scenarios. The SVM classifier consistently outperformed others.

TABLE IV: Accuracy Results for Key Scenarios

Scenario ID	Configuration	Accuracy
1. Device	Server	0.99
	Desktop	0.98
	IoT	0.93
2. Profit Strategy	Aggressive (100%)	0.98
	Robust (50%)	0.87
	Stealthy (10%)	0.91
3. Type	In-Browser	0.95
	Host-Based	0.99
4. Network	Fully Compromised	0.98
6. Network	Single Compromised (IoT)	0.94
9. Transferability	Binary In-Browser	0.99

B. Key Findings

- **Training Efficiency:** The model achieved 99% accuracy with only one hour of training data.
- **Stealth Analysis:** Stealthy strategies (10% throttle) reduced accuracy but remained detectable (¿90%) due to the distinct traffic patterns of mining (regular small packets) versus benign browsing.
- **Device Impact:** Server-based mining is easier to detect (99%) than IoT-based mining (93%), likely due to higher traffic volumes generated by powerful CPUs.
- **Transferability:** The model successfully detected malware on the WebOS TV even when trained only on Raspberry Pi data, proving the features are device-agnostic.

C. Resultados reprodução do artigo de referência

The reproduction of the base article can be obtained at https://github.com/edilsoneasj/cryptojackingReproduction. On the end of the output of the notebook cells there is the amount of time in minutes that took to execute the respective code section

5 Jupyter notebooks are available, corresponding to the respetive original code files. The original code had some errors with manipulation of dataframes, that were solved. 4 of the 5 codes were reproduced, however the 5th one named: ”Non-default parameters” takes aproximately 20 days to be reproduced by the authors. The code did not execute on GPUs, only CPUs, and took far more time. The authors used an Intel Xeon with 192GB of ram, and still took around 20 days (even using the cluster from CIN-UFPE would reset the job) to execute this last file, and that is the reason why I was not able to complete the full reproduciton. The authors use a 5-fold Cross-validation approach, and therefore the results obtained were very similiar, as can be seen from the jupyter notebook files. Only the time that I obtained to execute the codes was around 2x the amount the authors took. There were too many results to compare with the base article and therefore, I did not have enough time to properly put them here on tables.

D. Resultados proposta de melhoria do artigo de referência (CASO FEITA)

VIII. Conclusões e trabalhos futuros

A. Conclusion

The article successfully proposes a lightweight, network-based detection mechanism for IoT cryptojacking. It overcomes the limitations of hardware-locked IoT devices by analyzing network traffic metadata. With an accuracy of up to 99% using SVM and requiring only minimal training data, the solution is effective against various attacker strategies (stealthy, aggressive) and heterogeneous network environments.

B. Limitations and Open Problems

- **Encryption and Spoofing:** While the system works on metadata (packet size/time), heavy obfuscation or padding by advanced malware could potentially reduce feature distinctiveness.
- **IP Identification:** The current work detects *that* a device is mining but notes that identifying the specific malicious server IP is challenging due to asymmetric communication (12% server packets vs 88% client packets) and dynamic IPs.

C. Future Work

- **Malicious IP Identification:** Further investigation is required to pinpoint the malicious destination IPs despite the asymmetric traffic and use of proxies.

- **Real-time Deployment:** While the mechanism is ”lightweight,” deploying it on consumer-grade routers with very limited processing power requires further optimization for real-time inference.

REFERÊNCIAS

[1] Tekiner, E., Acar, A. & Uluagac, A. A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks.. *NDSS*. (2022)