Poster: Adversarial Perturbation Attacks on the State-of-the-Art Cryptojacking Detection System in IoT Networks

Kiho Lee
Sungkyunkwan University
Republic of Korea
kiho@skku.edu

Sanghak Oh
Sungkyunkwan University
Republic of Korea
sanghak@skku.edu

Hyoungshick Kim
Sungkyunkwan University
Republic of Korea
hyoung@skku.edu

ABSTRACT

The popularity of cryptocurrency raised a new cyber security threat dubbed *cryptojacking* representing malicious activities for abusing victims' computing resources without their consent to mine cryptocurrency. Recently, Tekiner *et al.* [1] proposed an effective cryptojacking detection technique using a machine learning model with the statistical properties of the network traffic for cryptojacking in the Internet of Things (IoT) devices. In this paper, however, we demonstrate that this state-of-the-art method can effectively be evaded by maliciously manipulating the network packets for cryptojacking. Our evaluation results show that packet manipulations (packet splitting, dummy packet/payload insertion, and a proxy network) can effectively evade the model's detection – the packet splitting technique significantly decreased the F1-score of the detection model from 0.93 to 0.30. Finally, the best combination of those packet manipulations can decrease the F1-score of the detection model to 0.21.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

Cryptojacking; IoT; Adversarial example; Machine learning

ACM Reference Format:
Kiho Lee, Sanghak Oh, and Hyoungshick Kim. 2022. Poster: Adversarial Perturbation Attacks on the State-of-the-Art Cryptojacking Detection System in IoT Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3548606.3563530

1 INTRODUCTION

Cryptocurrency is a digital asset designed to work as a medium of exchange to securely transfer and verify its ownership using cryptographic operations. In most public blockchain systems, cryptocurrency creation is regulated by a process dubbed *mining* which is defined as the computational process of validating transactions on a blockchain system. In public blockchain systems, users are encouraged to voluntarily participate in mining (requiring computational resources) by incentivizing a certain number of cryptocurrency assets as a reward for a user who completes the mining process successfully. Due to this economic incentive for mining, a new type of cyber threat dubbed "*cryptojacking*" was introduced.

Cryptojacking is the unauthorized use of a victim's computing resources without consent to mine cryptocurrency for an attacker [2]. There are many techniques for cryptojacking [3]: attackers compromise websites to install crypto-mining scripts or inject crypto-mining scripts into victims' network traffic to hijack victims' computing resources stealthily. Cryptojacking has also been performed to target IoT devices [4, 5].

Several defense methods have been proposed to mitigate cryptojacking attacks. For example, Ahmad *et al.* [6] presented a lightweight classification model to detect cryptojacking attacks on IoT devices. Recently, Tekiner *et al.* [1] proposed the state-of-the-art cryptojacking detection technique using a machine learning model with the statistical properties (e.g., packet length) of the network traffic for cryptojacking in IoT networks. The proposed model would be effective – it could obtain accuracy as high as 97% with only one hour of training data. However, in this paper, we show that this state-of-the-art cryptojacking detection model may not be sufficiently robust against adversarial examples by manipulating the network packets for mining intentionally through some perturbations. We used packet splitting, dummy packet/payload insertion, and a proxy network for perturbations to evade the detection of the model without degrading the effectiveness of the cryptojacking. Our main contributions are summarized as follows:

(1) We introduce new adversarial perturbation attacks to effectively evade the state-of-the-art cryptojacking detection system [1] in IoT networks. The source code of the attack implementation is available (https://github.com/SKKU-SecLab/MiningPerturbation) to foster further research for cryptojacking detection.
(2) We evaluate the attack performance of adversarial perturbation attacks through experiments in real-world network configurations. The experimental results show that the best perturbation combination can significantly decrease the system's detection accuracy (F1-score) by 0.72.

2 OVERVIEW OF TEKINER *ET AL.*'S METHOD

Tekiner *et al.* [1] recently proposed the state-of-the-art cryptojacking detection system in IoT networks, using the IoT devices' network traffic for cryptojacking. The proposed solution first extracts time-series features from network packet data between an IoT device and a mining server and analyzes their statistical properties to use them as the features for a classifier. In their implementation, the tsfresh package (https://tsfresh.readthedocs.io/en/latest/) was used to calculate the time-series features.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3563530

Tekiner *et al.* performed experiments to find the key features, the most accurate classifier, and the optimum training size and evaluated the effectiveness of their cryptojacking detection mechanisms under various attacker configurations and network conditions. The experimental results showed that the best classifier achieved 97% detection accuracy with only one hour of training data.

This paper shows that the performance of this detection system can significantly be degraded against adversarial examples using perturbations on network packets.

3 THREAT MODEL

We assume that a victim's IoT device is connected to a website that an attacker compromises to use the IoT device's computing resources to mine cryptocurrency. The attacker's goal is to evade the detection of a cryptojacking detection tool while performing cryptojacking on the IoT device.

In this paper, we consider two possible situations where the attacker can manipulate the device network traffic for cryptojacking. In the first situation, we assume that the victim's device is connected to the attacker's proxy server. Users can often use a proxy server for several reasons (e.g., accessing a server in a foreign country). Also, the attacker can trick users into running a script to set up the use of the attacker's proxy server. In the second situation, we assume that the attacker's malware is installed on the victim's device. In both cases, the attacker can monitor all network traffic from the victim's device, and the attacker can inject dummy network packets and modify the existing packet payload if needed.

4 PERTURBATIONS FOR ADVERSARIAL EXAMPLES

This section explains how to generate adversarial examples to evade the detection of Tekiner *et al.*'s method described in Section 2 while successfully performing cryptojacking on the IoT device. To generate adversarial network packets, we recommend the following four types of perturbations to manipulate network packets because they do not affect the functionality of crypto mining protocols at the application level.

(1) **Dummy packet** is to randomly insert a dummy packet into the network packets for crypto mining at t time interval.
(2) **Padding** adds k zero bytes at the end of a chosen packet to adjust its size and updates the packet length and checksum fields after padding.
(3) **Splitting** is to break a packet into several smaller packets. Given a sequence of packets for either TCP or UDP traffic, an attacker assembles the packets and then splits them into k packets again.
(4) **Obfuscation (Obf.) proxy** is to use a network proxy server obfuscating the distribution of packet sizes and timing among packets.

5 EXPERIMENTS

This section describes the experiments to show the effectiveness of the adversarial examples crafted using the perturbations presented in Section 4 against Tekiner *et al.*'s detection system [1].

5.1 Experimental Setup

To manipulate network packets, we utilized *Scapy* (https://scapy.net/) and *Bettercap* (https://www.bettercap.org/). We also used the *Obfs4* proxy (https://github.com/Yawning/obfs4) as a representative network proxy server obfuscating network traffic patterns.

To show the feasibility of the adversarial examples generated in the real world, we used the Raspberry Pi 3 model B version 1.2 for IoT devices and a router for the attacker (see Figure 1). We used *Wireshark* (https://www.wireshark.org/) as a packet collector and analyzer. We collected normal network packets for web games and video streaming for two days (normal dataset); we collected the network packets for conventional Monero mining in a web browser for two days (cryptojacking dataset); we collected the network packets for Monero mining through perturbations (perturbated cryptojacking dataset). For Monero mining, we configured the device to use 100% CPU and 50% in a mixed manner. We built the classifier with 50% of the normal dataset and 50% of the cryptojacking dataset. The remaining datasets were used for testing.

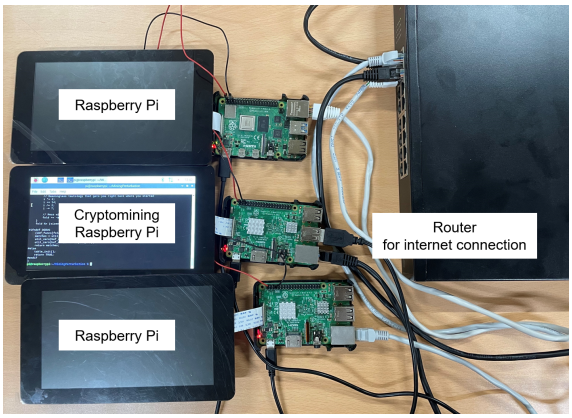

Figure 1: Three Raspberry Pi devices (One of three for cryptomining) and a router used for experiments.

As presented in Section 4, we applied perturbations to generate adversarial examples to evade the detection of the classifier. We utilized *Bettercap* to capture the network packets between the crypto mining server and the IoT devices. We also utilized *Scapy* to manipulate the network packets using the perturbations presented in Section 4. For dummy packet insertion, we randomly adjusted the transmission interval with the *sleep()* function to send dummy packets such as PSH, ACK, and SYN until crypto mining was terminated. Padding changes the size of each packet by adding k zero bytes to the request packet of the crypto mining server. For splitting, we split the packets transmitted from the IoT device to the crypto mining server when each packet size is more than the maximum size we have set, using the *fragment()* function of the *Scapy*. We also deployed the *Obfs4* proxy server to obfuscate the network traffic for crypto mining.

We examined various parameter values for each attack type to find the optimal parameters in perturbations to increase their evasiveness against the state-of-the-art cryptojacking detection system [1]. For the dummy packet perturbation, we analyzed the effectiveness of dummy packets transmitted by varying the transmission interval from 1 to 60 seconds. For the padding, we analyzed

Table 1: Performance of the detection model against perturbations.

Perturbation	Precision	Recall	F1-score
Baseline	0.93	0.93	0.93
Dummy packet	0.25	0.50	0.34
Padding	0.58	0.51	0.36
Splitting	0.22	0.48	0.30
Obf. proxy	0.24	0.49	0.33
Dummy packet & Padding	0.24	0.49	0.33
Dummy packet & Splitting	0.24	0.48	0.32
Dummy packet & Obf. proxy	0.22	0.47	0.30
Padding & Splitting	0.25	0.50	0.34
Padding & Obf. proxy	0.24	0.49	0.33
Splitting & Obf. proxy	0.19	0.43	0.26
Dummy packet & Padding & Splitting	**0.15**	**0.38**	**0.21**
Dummy packet & Padding & Obf. proxy	0.22	0.47	0.30
Dummy packet & Splitting & Obf. proxy	0.17	0.42	0.24
Padding & Splitting & Obf. proxy	0.19	0.43	0.26
Combining all the four perturbations	0.20	0.45	0.28

the effectiveness of zero padding length by varying the zero padding length from 1 byte to 30 bytes. For the splitting perturbation, we controlled the maximum packet size with the maximum transmission unit (MTU). We conducted an initial study. Our initial experimental results suggested four seconds for the dummy packet perturbation, four bytes for the padding perturbation, and 500 bytes for the MTU in the splitting perturbation.

5.2 Evaluation

We evaluate the effectiveness of the perturbations with respect to three weighted average metrics: precision, recall, and F1-score, which are used in the baseline model. In the case of weighted average, the performance metrics are weighted accordingly:

$$score_{weighted-avg} = W_1 \cdot score_{class0} + W_2 \cdot score_{class1} \quad (1)$$

The experimental results show that all the perturbations effectively decreased the model's detection accuracy (see Table 1). Splitting is the most effective perturbation technique, achieving a drop of 0.63 in the F1-score. The use of the *Obfs4* proxy server is the second most effective one, achieving a drop of 0.6 in the F1-score. Dummy packet and padding also significantly reduced the F1-score of the model from 0.93 to 0.34 and 0.36, respectively. To find the best perturbation combination, we analyze the evasiveness of each combination of multiple perturbations and found that some combinations are more effective than a single method alone. The best combination of perturbations is to apply all the perturbations together except for the obfuscation proxy technique, achieving a drop of 0.72. We surmise that the effects of the obfuscation proxy technique may be redundant when using the padding technique because the *Obfs4* proxy server uses its own padding scheme.

5.3 Discussion

Tekiner *et al.*'s method [1] extracts time-series features from network packet data between an IoT device and a mining server and analyzes their statistical properties (e.g., packet size and time intervals between packets) to use them as the features for a classifier. Although the proposed model can effectively detect conventional crypto mining network traffic, we found that the model's detection accuracy can significantly be degraded when network packets are slightly changed with some simple perturbations. Therefore, building a robust classifier to fight against adversarial examples with such perturbations is necessary.

Even though the dummy packet insertion technique can simply be implemented, it inherently incurs the communication cost of transmitting some unnecessary packets periodically. Applying the padding technique would be a better option in terms of communication overhead. Although we need to update the packet length and checksum fields to fix the checksum error, this perturbation does not require new network packets. The packet splitting technique achieves the best evasiveness results compared to the other perturbations. Each packet is split into several smaller packets to hide the distribution of the network packets for crypto mining. However, the packet splitting process may incur significant time overhead for generating new packets. Using the *Obfs4* proxy server, 'zeroPadBytes' is appended to each packet to have a size equal to the MTU, and the packet is transmitted in an encryption form. Therefore, the packets transmitted through the *Obfs4* proxy consistently have the maximum size.

6 CONCLUSION

In this paper, we introduce simple yet effective techniques to generate adversarial examples to evade the detection of a state-of-the-art network-based detection system in IoT networks. We suggest the four practical perturbation techniques. To evaluate the effectiveness of those perturbations, we configured a real-world IoT network testbed and generated adversarial examples in real-time. The experimental results show that our perturbations are significantly effective in decreasing the model's detection accuracy. The best perturbation combination can decrease the F1-score of the detection model from 0.93 to 0.21.

As part of future work, we will develop more advanced perturbation techniques to generate adversarial examples to mimic benign network application traffic. Also, we will extend our evaluation to other cryptojacking detection methods for generalization.

7 ACKNOWLEDGEMENT

The authors would thank anonymous reviewers. Hyoungshick Kim is the corresponding author. This work was supported by Korea Internet & Security Agency (KISA) grant (1781000003) and Institute for Information & communication Technology Promotion (IITP) grant (No. 2019-0-01343).

REFERENCES

[1] Ege Tekiner, Abbas Acar, and A Selcuk Uluagac. A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks. In *Proc. of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2022.
[2] Ege Tekiner, Abbas Acar, A Selcuk Uluagac, Engin Kirda, and Ali Aydin Selcuk. SoK: Cryptojacking Malware. In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021.
[3] Hugo LJ Bijmans, Tim M Booij, and Christian Doerr. Inadvertently Making Cyber Criminals Rich: A Comprehensive Study of Cryptojacking Campaigns at Internet Scale. In *Proc. of the USENIX Security Symposium (USENIX Security)*, 2019.
[4] Elisa Bertino and Nayeem Islam. Botnets and Internet of Things Security. *Computer*, 2017.
[5] Larry Greenemeier. How Cryptojacking can Corrupt the Internet of Things, 2018. URL: https://www.scientificamerican.com/article/how-cryptojacking-can-corrupt-the-internet-of-things/.
[6] Azuan Ahmad, Wan Shafiuddin, Mohd Nazri Kama, and Madihah Mohd Saudi. A New Cryptojacking Malware Classifier Model Based on Dendritic Cell Algorithm. In *Proc. of the International Conference on Vision, Image and Signal Processing (ICVISP)*, 2019.