Poster: Code-Reuse Attack Detection Using Sequential Hypothesis Testing in IoT*

Jun-Won Ho
Department of Information Security
Seoul Women's University
Seoul, South Korea
Email: jwho@swu.ac.kr

SeungYeon Kang, SeJin Kim†
Department of Information Security
Seoul Women's University
Seoul, South Korea
Email: {swuifksy, 2015111503}@swu.ac.kr

Abstract

As in conventional computing systems, IoT devices are vulnerable to different types of attacks. Among a variety of possible attacks against IoT devices, code-reuse attacks are dangerous in the sense that attacker can easily take malicious benefits from reusing the normal codes in IoT devices and it is not easy to detect these attacks. In order to protect IoT ecosystem from code-reuse attacks, we propose a statistical scheme to detect code-reuse attacks by scrutinizing the packets incoming into IoT devices. Through simulation study, we show that our proposed scheme efficiently and resiliently detects code-reuse attacks in IoT.

1. Introduction

IoT devices have substantially less computation, communication, and storage capability than conventional computing systems. This is because low-priced devices are usually employed to provide computation, sensing, and communication functionality in IoT ecosystem. However, IoT devices are similarly vulnerable to diverse attacks that could be launched against conventional computing systems. Thus, IoT security mechanisms should be as simple and efficient as possible while providing robust security resilience against various attacks with low overheads.

Among diverse attacks against IoT ecosystem, code-reuse attacks [4] are very detrimental in the sense that adversary could put malicious devastation into IoT devices by exploiting vulnerabilities (e.g. buffer overflow vulnerabilities) of these devices and craftily reusing the normal instructions in them. Many researchers proposed various schemes to thwart code-reuse attacks in diverse types of systems. In [1], code randomization technique is applied for code-reuse attack defense. In [2], code-reuse attacks are prevented by validating the target addresses of all instructions that are discerned as potential attack points through binary executable analysis in IoT devices. In our previous work [3], we detect code-reuse attacks by applying Kullback-Leibler divergence to the packets received and generated/sent by IoT devices such that these packets contain instruction addresses. However, these schemes need to repeatedly randomize the address space layout of instructions or perform verification process on the target addresses or inspect all incoming and outgoing packets, and thus they are not adequate for meeting the aforementioned IoT security requirements.

To soothe these limitations, we propose a code-reuse attack detection scheme based on the Sequential Hypothesis Testing (SHT) [5]. Specifically, each IoT device feeds each incoming packet into the sequential hypothesis testing and detects code-reuse attacks by harnessing the intuition that the addresses of the instructions are highly likely included in incoming packets in case of code-reuse attacks. The simulation results demonstrate that our proposed scheme attains at least 98.5% detection rate with at most 10.1 samples on an average, being suitable for the above IoT security requirements.

2. SHT-based Detection of Code-Reuse Attacks in IoT

We first define *code-reuse evidence* as the address of the instruction placed in code region of IoT device.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1C1B1014126).

† *These two authors contributed equally to this work.*

Upon receiving a sequence of packets from other IoT devices, every IoT device makes a decision on whether these incoming packets contain code-reuse evidence or not. We employ the Sequential Hypothesis Testing (SHT) [5] for this decision making process. In the SHT, we define null hypothesis (H_0) and alternate hypothesis (H_1), where H_1 (resp. H_0) represents that a sequence of incoming packets does (resp. not) contain code-reuse evidence.

We split a sequence of packets incoming into a 32-bit (resp. 64-bit) IoT device into a series of 4-byte (resp. 8-byte) blocks and denote them by $L_1, L_2, \ldots, L_i, \ldots (i \geq 1)$. Furthermore, U_i is defined as a Bernoulli random variable such that $U_i = 1$ if the value of L_i falls within the address space of instructions in code region of IoT device. Otherwise, $U_i = 0$. Given the success probability s of the Bernoulli distribution, we have $s = \Pr(U_i = 1) = 1 - \Pr(U_i = 0)$. Then, thresholds s_0 and s_1 ($s_0 < s_1$) are preset in the SHT such that the condition of $s \geq s_1$ (resp. $s \leq s_0$) contributes to the acceptance of H_1 (resp. H_0).

Given that U_i is considered to be a sample, the log-probability ratio T_k on $k (k \geq 1)$ samples is calculated as $T_k = \ln \frac{\Pr(U_1,\ldots,U_k|H_1)}{\Pr(U_1,\ldots,U_k|H_0)}$. Let α' and β' denote a user-configured false positive rate and a user-configured false negative rate, respectively. If $T_k \geq \ln \frac{1-\beta'}{\alpha'}$, the SHT accepts H_1 and IoT device removes the blocks contributing to H_1 acceptance from the system. If $T_k \leq \ln \frac{\beta'}{1-\alpha'}$, the SHT accepts H_0. Otherwise, it continues the testing process. If the SHT accepts H_0 or H_1, it restarts the testing process with new samples.

To provide resilience against intelligent attacks, we consider random sampling strategy in which the SHT randomly determines samples in such a way that the ith sample U_i is obtained from the corresponding ith block L_i with probability ϵ. Under this sampling strategy, attacker will likely randomly determine the blocks containing code-reuse evidence, resulting in independence of samples. Furthermore, samples are generally independent of each other in benign case. Hence, it is reasonable that U_i is assumed to be independent and identically distributed. Under this i.i.d. assumption, when N_k is defined as the number of times that $U_i = 1$ in the k samples, the SHT can be restated as follows: If $N_k \leq \frac{\ln \frac{\beta'}{1-\alpha'} + k \ln \frac{1-s_0}{1-s_1}}{\ln \frac{s_1}{s_0} - \ln \frac{1-s_1}{1-s_0}}$, the SHT accepts H_0. If $N_k \geq \frac{\ln \frac{1-\beta'}{\alpha'} + k \ln \frac{1-s_0}{1-s_1}}{\ln \frac{s_1}{s_0} - \ln \frac{1-s_1}{1-s_0}}$, the SHT accepts H_1. Otherwise, the SHT goes on the testing process.

We evaluate the SHT with random sampling strategy through a simple simulation. We set $\alpha' = \beta' = 0.01$, $s_0 = 0.1$ and $s_1 = 0.9$. We also configure $\epsilon = 0.1, 0.15, 0.2$ and set the number of blocks to 500. Each block is chosen to have code-reuse evidence with probability $p = 0.5$ and the chosen block is marked as H_1 type. We take a sample corresponding to a block with probability $\epsilon = 0.1, 0.15, 0.2$. Our simulation results reveal that code-reuse attack detection rate fulfills at least 98.5% in all three configurations of ϵ. Furthermore, as shown in Figure 1, the SHT reaches H_1 decision with at most 10.1 samples on an average in all three configurations of ϵ. Additionally, we identify that a growth in ϵ contributes to a dwindle in detection time. From these observations, we discern that our SHT-based scheme accomplishes high detection capability with little overhead.

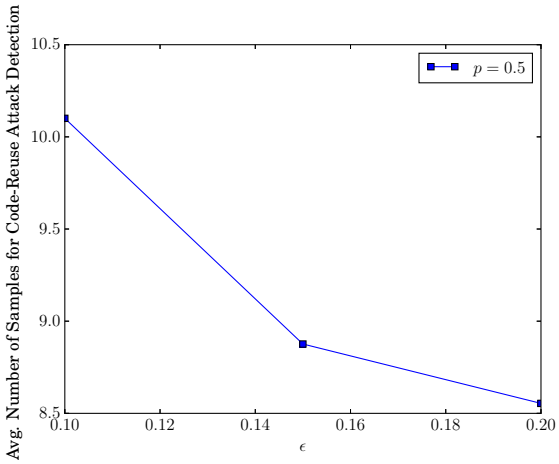

Figure 1. Average number of samples required for the SHT to detect code-reuse attacks.

References

[1] S. Crane, C. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, M. Franz. Readactor: Practical Code Randomization Resilient to Memory Disclosure. In *IEEE S&P*, 2015

[2] J. Habibi, A. Panicker, A. Gupta, and E. Bertino. DisARM: Mitigating Buffer Overflow Attacks on Embedded Devices. In *CERIAS Tech Report*, 2015-15, 2015.

[3] J. Ho. Code-Reuse Attack Detection Using Kullback-Leibler Divergence in IoT. In *IJASC Vol 5. No.4 54-56*, 2016.

[4] R. Roemer, E. Buchanan, H. Shacham, and S. Savage. Return-oriented programming: Systems, languages, and applications. In *ACM Transactions on Information and System Security*, 15, 1 (Mar. 2012), 2:1-2:34.

[5] A. Wald. Sequential Analysis. *Dover Publications*, 2004.