POSTER: Static ROP Chain Detection Based on Hidden Markov Model Considering ROP Chain Integrity

Toshinori Usui[1], Tomonori Ikuse[2], Makoto Iwamura[1], Takeshi Yada[1]

[1]NTT Secure Platform Laboratories
3-9-11 Midori-cho, Musashino-shi
Tokyo, Japan
{usui.toshinori, iwamura.makoto, yada.takeshi}@lab.ntt.co.jp

[2]NTT Security (Japan) KK
1-5-2 Higashi-shinbashi, Minato-ku
Tokyo, Japan
tomonori.ikuse@ntt.com

ABSTRACT

Return-oriented programming (ROP) has been crucial for attackers to evade the security mechanisms of operating systems. It is currently used in malicious documents that exploit viewer applications and cause malware infection. For inspecting a large number of commonly handled documents, high-performance and flexible-detection methods are required. However, current solutions are either time-consuming or less precise. In this paper, we propose a novel method for statically detecting ROP chains in malicious documents. Our method generates a hidden Markov model (HMM) of ROP chains as well as one of benign documents by learning known malicious and benign documents and libraries used for ROP gadgets. Detection is performed by calculating the likelihood ratio between malicious and benign HMMs. In addition, we reduce the number of false positives by ROP chain integrity checking, which confirms whether ROP gadgets link properly if they are executed. Experimental results showed that our method can detect ROP-based malicious documents with no false negatives and few false positives at high throughput.

CCS Concepts

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

Keywords

Return-Oriented Programming; Attack Code Detection; Hidden Markov Model

1. INTRODUCTION

Malicious documents that exploit vulnerabilities are used for targeted attacks. To detect malicious documents, as Stancill et al.[7] proposed, it is effective to focus on return-oriented programming (ROP)[6]. ROP detection is mainly achieved based on dynamic analysis. However, dynamic detection methods are generally time-consuming due to the execution duration of viewer applications for analysis and are not suitable for inspecting a large number of documents.

Several fast static methods have recently been proposed. Check My Profile[7] takes memory snapshots and analyzes them by profiling ROP gadgets and chains. n-ROPdetector[8] collects gadgets from the Metasploit framework and detects based on their pattern matching in networks. STROP[9] uses several rules for detecting ROP chains in network packets. eavesROP[3] applies the fast Fourier transform (FFT) for effective pattern matching of ROP chains. These methods achieve fast detection and are suitable for inpecting a large number of documents. However, they are less flexible regarding changes in attack trends or attackers' evasion when rules or thresholds are publicly known. This is because these methods mainly depend on a pattern matching-based or rule-based method. Therefore, to keep tracking changes in attack trends, more flexible methods are required.

In this paper, we propose a novel learning-based method of statically detecting ROP chains in malicious documents. The method adopts statistical learning with HMMs[4] for modeling and a quick probability calculation-based algorithm for detection. Due to this, our method is flexible and inspects rapidity. We also introduce ROP chain integrity (RCI) checking, which checks whether the chain of ROP gadget addresses links properly. By considering the RCI, we can improve the accuracy of the detecting ROP chains. Experimental results suggest that our method can detect the ROP chains in malicious documents with high precision in a short time.

2. METHOD

Figure 1 gives an overview of our method. The method adopts statistical learning using HMMs to detect ROP chains. We first explain the design of HMMs and the way our method detects with them. Then, to reduce the number of false positives, we use RCI checking for improving detection.

2.1 HMM Design

For modelling ROP-based malicious documents, an HMM is designed to use the byte sequence of documents as an observed sequence, while the label sequence as a hidden state sequence. Therefore, emission symbols of the HMM are the set of 0x00-0xFF. The ROP chains generally consist of three components, ROP gadget addresses (addr), constant arguments (const) and junk codes (junk). Therefore, the state spaces are the set of addr[1-4], const[1-4], junk and doc la-

CCS'16 October 24-28, 2016, Vienna, Austria

ACM ISBN 978-1-4503-4139-4/16/10.

DOI: http://dx.doi.org/10.1145/2976749.2989040

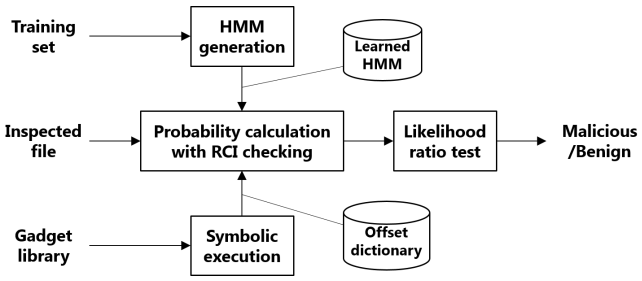

Figure 1: Overview of method

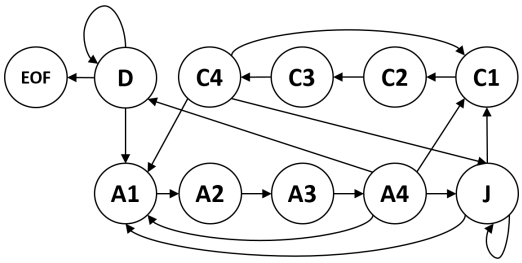

Figure 2: Transition diagram for byte-wise HMM of 32-bit ROP chain

bels, where the index number indicates the byte-wise position in the components (e.g., addr3 means the third byte of a ROP gadget address). Figure 2 depicts the state transition diagram of an HMM of 32-bit ROP chains designed for our detection method. In the diagram D denotes doc, A denotes addr, C denotes const and J denotes junk.

With our method, the HMM model parameters $\theta = (A, B, \pi)$ are generated by supervised learning. We apply labeled documents for the training data every byte of which has a corresponding label. By using the training data, the transition probability $a_{i,j} \in A$ that the transition from state i to state j, the emission probability $b_{j,o} \in B$ that state j emits symbol o, and the initial state probability $\pi_i \in \pi$ of state i are computed as follows;

$$a_{i,j} = \frac{K_{i,j}}{\sum_{k \in Z} K_{i,k}}, b_{i,o} = \frac{M_{i,o}}{\sum_{p \in V} M_{i,p}}, \pi_i = \frac{N_i}{\sum_{j \in Z} N_j} \quad (1)$$

where V is the set of emission symbols, Z is the set of hidden states, $K_{i,j}$ is the number of transitions from state $i \in Z$ to state $j \in Z$, the $M_{i,o}$ is the number of the emitted symbols $o \in V$ by the state i, and N_i is the number of initial states i.

While calculating the emission probabilities of addr[1-4], a sampling bias problem occurs. This is because the gadget addresses that appear in the known samples are quite limited. However, attackers can create ROP chains that have equivalent behavior to the known chains by using addresses that do not exist in the known chains. Therefore, we avoid this problem by learning the libraries adopted for ROP gadgets. We extract the gadget address candidates from the library used to create chains. The extracted addresses are applied for learning the emission probabilities of addr[1-4] by Equation (1).

We applied HMMs because of the following reasons. First, the documents are regarded as sequence data, in which structured learning methods such as an HMM are suitable. Second, the relation between observed bytes and ROP component labels is similar to latent variable models such as an HMM. Third, the transitions between two of the ROP components and emissions of bytes have convergence properties.

2.2 ROP Chain Integrity

Static ROP chain detection sometimes causes false positives due to the appearance of gadgets address-like byte sequences in the data stream. For evading the occurance of these false positives, we introduce the concept of RCI checking to static ROP chain detection. The RCI is used to evaluate the integrity of a ROP gadget properly linking to another ROP gadget. If gadgets do not link properly, the chain is considered an invalid ROP chain. We call this situation chain violation (CV). By RCI checking, we can reduce the number of false positives derived due to accidentally happened gadget-address-like byte sequences appearing. This is because the false positives mostly cause CV.

For RCI checking between two consecutive gadget-address candidates, the change offset of the stack pointer when the gadget is executed is required. To obtain this, we previously executed all the gadget candidates in the library used for ROP chains. During the execution, we had to handle the branches that appeared in the gadget candidates. Hence, we leveraged symbolic execution. Through this, we produced a dictionary in which the gadget-candidate addresses and the corresponding stack pointer offsets are stored. Although this task takes a certain amount of time, the inspection does not get longer because it is done before inspection.

To adopt RCI checking in our probabilistic method, we computed the probability that the HMM emits the observed byte sequence with no CV. This is used as the likelihood of ROP-based malicious documents. The likelihood $L(\theta|X)$ is computed as follows;

$$L(\theta|X) = P(X, \bigcap_{(i,j) \in J_X} F_{i,j}|\theta) = P(X|\theta)P(\bigcap_{(i,j) \in J_x} F_{i,j}|X, \theta) \quad (2)$$

where X is the observed byte sequence, i, j are the steps that the corresponding byte $x_i, x_j \in X$ are interpreted as chain source and destination, J_X is the set of (i, j) in X and $F_{i,j}$ is a stochastic variable with which set (i, j) does not cause CV.

Since computing the probability above is quite difficult, we made two assumptions below for making computations easier. (i) The probability that a ROP gadget address does not cause CV is independent of the probability that the other ROP gadget addresses do not cause CV. (ii) The state probability of the chain source is independent of that of the chain destination. By assuming these, the probability of Equation (2) is approximately calculated as follows;

$$L(\theta|X) \approx P(X|\theta) \prod_{(i,j) \in J_x} 1 - P(i = A1|X, \theta)P(j \neq A1|X, \theta) \quad (3)$$

where $A1$ is the label of addr1.

Here, $P(X|\theta)$ and $P(i = \cdot|X, \theta)$ are quickly calculated using the forward and forward-backward algorithms[4], respectively. Therefore, we can also compute the entire likelihood $L(\theta|X)$ in a short time.

2.3 Detection

Our method detects by conducting likelihood ratio test. Hence, the method first calculates the likelihood ratio Z as follows;

$$Z = \frac{P(X|H_{Mal})}{P(X|H_{Ben})} = \frac{L(\theta_{Mal}|X)}{P(X|\theta_{Ben})} \quad (4)$$

where H_{Mal} is the hypothesis that the inspected document is malicious, H_{Ben} is the hypothesis that the inspected document is benign, θ_{Mal} is the HMM of malicious documents with ROP chains, and θ_{Ben} is the HMM of benign documents. Then, if $Z > t$ the document is detected as malicious; otherwise, it is benign, where t is a threshold. The order of computing Z is $O(N)$ where N is the length of the observed byte sequence ($N = |X|$). Thus, the detection of our method is quick.

3. EXPERIMENT

We conducted experiments to evaluate our method. Table 1 lists the datasets used in the experiments. As malicious samples, we collected the malicious documents that exploit CVE-2014-1761 that is new and widely used in the wild. We confirmed that the malicious samples included three different ROP chains. Since CVE-2014-1761 samples use the library file of MSCOMCTL.OCX for their ROP chain, the HMM is also trained by the library.

Table 1: Datasets

Category	Label	Samples	Source
Training set	Malicious	1	Metasploit[5]
	Benign	225	Garfinkel et al.[1]
Test set	Malicious	40	VirusTotal[2]
	Benign	900	Garfinkel et al.[1]

Using the datasets, our method first generated the HMMs of malicious and benign documents with the training set then inspected the test set with the HMMs. The threshold t is defined by 5-fold cross validation in the test set. During the process, we chose the best t within a range that does not cause false negatives. The experiments were conducted on the following environment; CPU: Intel Xeon CPU E5-2660 v3 @2.60GHz, Memory: 32GB and OS: Ubuntu 14.04 LTS. All inspections were done on a single CPU.

The experimental results suggest that our method detected all malicious samples with no false negatives and the average false positive rate was 3%. During the experiments, our method inspected in 0.4 s/file on average and its throughput was around 1.73 Mbps/CPU.

4. DISCUSSION AND CONCLUSION

We first discuss two limitations of our method. The first is that since the method is based on static analysis, detection is limited to the ROP chains visible on the byte sequence. Namely, ROP chains dynamically generated by scripts, such as JavaScript in a PDF, are out of our scope. Therefore, we mainly focused on OLE- or RTF-formatted documents. The second limitation is that during RCI checking, the gadgets that caused the ambiguous stack pointer offset (e.g., gadgets that ended with jmp [eax] without setting the eax register) were excluded in this research.

We also discuss the intervals of the HMM model updates. A model update is required depending on the update of the library used for the gadgets. A library update is rarely done on non-ASLR (address space layout randomization) libraries that are usable for ROP chains. Also, a model update may be required when the structure of the ROP chains is changed drastically (e.g., appearance of ROP chains with numerous junk codes). However, as far as we know, this is not observed in the wild. Hence, the model-update frequency is low with our method.

Although our method is only focused on ROP in this paper, the method is also effective against jump-oriented programming (JOP) and call-oriented programming (COP) depending on the implementation. Our method can be applied to data streams other than documents despite the focus of this paper.

In this paper, we proposed a method that statically detects ROP chains in malicious documents. Our method generates two HMMs and detects the ROP chains by conducting likelihood ratio test considering the RCI. The experimental results suggest that our method can detect ROP-based malicious documents with no false negatives and few false positives at high throughput. Improving the learning method, considering other HMM-based detection algorithms, and conducting larger-scale experiments are possible future work.

5. REFERENCES

[1] S. Garfinkel et al. Bringing science to digital forensics with standardized forensic corpora. *digital investigation*, 6:S2–S11, 2009.

[2] Google. Virustotal. https://www.virustotal.com/.

[3] C. Jämthagen et al. eavesrop: Listening for rop payloads in data streams. In *Proceedings of the International Conference on Information Security*, pages 413–424. Springer, 2014.

[4] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[5] Rapid7. Metasploit. http://www.metasploit.com/.

[6] H. Shacham. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 552–561. ACM, 2007.

[7] B. Stancill et al. Check my profile: Leveraging static analysis for fast and accurate detection of rop gadgets. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 62–81. Springer, 2013.

[8] Y. Tanaka et al. n-ropdetector: Proposal of a method to detect the rop attack code on the network. In *Proceedings of the 2014 Workshop on Cyber Security Analytics, Intelligence and Automation*, pages 33–36. ACM, 2014.

[9] C. YoungHan et al. Strop: Static approach for detection of return-oriented programming attack in network. *IEICE Transactions on Communications*, 98(1):242–251, 2015.