Journal of Information Security and Applications 42 (2018) 46–56

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

SWORD: Semantic aWare andrOid malwaRe Detector

Shweta Bhandari[a,*], Rekha Panihar[a], Smita Naval[b], Vijay Laxmi[a], Akka Zemmari[c], Manoj Singh Gaur[d]

[a] Malaviya National Institute of Technology Jaipur (MNIT Jaipur), India
[b] National Institute of Technology Warangal (NIT Warangal), India
[c] LaBRI - University of Bordeaux, CNRS, Talence cedex 33405, France
[d] Indian Institute of Technology Jammu, J&K, India

ARTICLE INFO

Article history:

Keywords:
Android
Malware analysis
Semantic analysis
Information leakage

ABSTRACT

Malicious android applications have become more advanced and severe threat to user privacy, confidentiality, integrity, money, and device. The process of malware evolution mainly consists of modifications to existing malware using repackaging of apps employing polymorphism, metamorphism and injecting malicious code. The existing dynamic approaches can handle polymorphism, metamorphism and repacking of apps but failed to address code injection at runtime, as it modifies the control/data flow. In this paper, we present a semantic aware dynamic malware detection tool, SWORD. It encapsulates the semantics of Android apps in such a way that makes it resilient towards injection-based evasion techniques. The intuition behind specifying the semantics of apps lies in applying Asymptotic Equipartition Property (AEP) inherited from information theory domain. The semantics of the app are captured using a sequence of system-calls. To assess the efficacy of SWORD, we carried out comprehensive experiments on 6000 execution traces of 2000 applications (1000 malware apps belonging to 119 different families and 1000 benign apps, selected randomly from 12,000 Google Play store apps). We obtain a detection accuracy of 94.2%. Moreover, we show that SWORD can cope with the code injection based evasion techniques.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

Smartphone has become quite important to our day to day life for social networking, online shopping, banking, *etc.* According to a study made by Statista [1], around 2.08 billion of the total population are smartphone users. Nowadays, Android is the most popular mobile operating system, with 85% of the worldwide smartphone sales to end users in 2017 [2]. It is becoming the key target for malware adversaries as it is open sourced and Android mobiles are available at very low cost. Although, the Android OS added 'app verification, a security feature,' yet banking trojans and other malicious android apps are actively spreading. Over the past few years, we are witnessing an upsurge in the events of malicious programs in the form Android applications (apps). For instance, Android malware like SlemBunk and Marcher actively target US financial institutions customers [3]. Therefore, to ensure security and privacy of an Android user, detection of malicious apps becomes primary line of defence [4,5].

Nowadays, detection of malicious apps is a challenging task as these apps are embedded with numerous evasion techniques, which put a question mark on the efficiency of existing detection mechanisms. The popular evasion techniques include repackaging of apps, anti-emulation, code-injection, polymorphism, metamorphism, and to name a few. Recent works [6–11] show that static approaches are not able to capture evasion-aware malicious apps. On the other hand, dynamic analysis operates by executing the program code. It is more precise as no approximation or presumption needs to be done [12]. Therefore, the effect of evasions like repackaging, obfuscation, metamorphism, and polymorphism can be invalidated by dynamic analysis technique [13]. However, these techniques are completely dependent on the runtime sequences of events.

To evade existing malware detection techniques, the malware authors try to inject irrelevant or independent code at runtime. There are mainly three ways [14] to inject code into already running Android application (1) using `DexClassLoader`, an Android app can invoke the classes' methods of any downloaded app during runtime, (2) invoking API named as `createPackageContext`, an android app can load and invoke resources (images, files, and codes) of other app [14] and (3) using OS shell [15], an application can redirect program control to some other code and also can

* Corresponding author.
E-mail address: er.shwetabhandari@gmail.com (S. Bhandari).

https://doi.org/10.1016/j.jisa.2018.07.003
2214-2126/© 2018 Elsevier Ltd. All rights reserved.

execute it. This injection alters the actual runtime sequence of events, therefore in this paper, we propose a dynamic malware detection technique named SWORD, that handles injection-based malicious apps. To capture the actual semantics of Android apps, we make use of system-call sequences because:

1. system-calls are independent of Android's compilation and running environment whether it is Dalvik virtual machine (DVM) or Android runtime (ART).
2. system-calls provide a gateway to access system-level services and are required to instigate malicious attacks such as premium calls, downloading other malicious apps, transferring bank credentials, etc [16,17].

We outline a proactive semantic detection model for capturing evasion-aware malicious Android apps. However, we do not claim that our approach can detect every class of evasion-aware malware. Our approach is only capable of detecting malicious apps that inject system-call at the runtime to evade the detection as well as non-evasive apps. SWORD employs AEP property widely deployed in the domain of information theory. This property states that "there are few paths of a graph that concentrates almost all information of the program under analysis". To apply AEP, we extract system-call traces by extending QEMU process with strace and Monkey tool [18] is used to stimulate the app. We established the resemblance of system-call traces as the ergodic Markov chain so that we can apply the phenomenon of AEP. We transform the system-call sequences into sequential system-call graph (SSG). Then, we find typical paths from SSG. Using typical paths, we construct our learning and detection model. In summary, we make the following contributions:

1. We design our model on the top of QEMU [19] and Monkey tool [20] to automatically capture the run-time behavior of an Android app in terms of system-call sequences.
2. We prove that acquired system-call traces represent the ergodic Markov chain and therefore we can apply the AEP concept to construct our semantic detector.
3. We compute the "typical paths" and use average logarithmic branching factor (ALBF) of paths to create our feature vector. Our feature vector is constructed using histogram binning technique. Further, we verify our semantic model using test samples to differentiate malicious and benign Android apps.
4. We assess the resiliency of our approach by injecting thousands of irrelevant calls into the traces of malicious Android apps. We observe that the proposed approach sustained the system-call injection attack.

The rest of the paper is organized as follows. We start by introducing the background for the proposed approach in Section 2. We present the proposed work in Section 3. In Section 4, we show the experimental setup and results to evaluate the proposed approach. In Section 5, we enumerated some of the limitation possess by SWORD. The related work is discussed in Section 6. Finally, we give concluding remarks to our approach in Section 7.

2. Background

We, now present some background on the ergodic markov chain and AEP as both of these concepts are borrowed from information theory domain. This will give better readability to our work.

2.1. Ergodic markov chains

Definition 1. A *Markov chain* is a discrete-time stochastic process $(X_t)_{t\geq 0}$ s.t. each random variable X_t takes values in a discrete set S, called *space state*, and for any s, s' and $s_0, s_1, \ldots, s_{t-1} \in S$,

$$\mathbb{P}r\big(X_{t+1} = s \mid X_t = s', X_{t-1} = s_{t-1}, \ldots, X_0 = s_0\big) = \mathbb{P}r\big(X_{t+1} = s \mid X_t = s'\big). \quad (1)$$

If the set S is finite then the chain is said to be *finite-state*.

Remark 1. Eq. (1) is called *memoryless* property and it simply means that, as time goes by, the process loses the memory of the past.

The chain is characterized by the space state S and by its *transition matrix* $P = \big(p_{i,j}\big)_{(s_i,s_j)\in S\times S}$, where,

$$p_{i,j} = \mathbb{P}r\big(X_{t+1} = s_j \mid X_t = s_i\big), \forall t \geq 0, \text{ and } \forall (s_i, s_j) \in S \times S. \quad (2)$$

Note that the transition matrix P verifies two properties: (1) its elements are all positive, and (2) each row sums to 1.

It is always possible to represent a finite-state Markov chain by a *transition graph* $G = (S, \tau)$ where S is the space state and τ corresponds to the transition matrix: for any pair of states s_i and s_j in S, $(s_i, s_j) \in \tau$ if and only if $p_{i,j} > 0$. The graph G is, thus, an oriented weighted graph. Given $t \geq 0$, the *distribution* at time t of the Markov chain is given by:

$$\pi_s^{(t)} = \mathbb{P}r(X_t = s), \forall s \in S.$$

To characterize the chain completely, in addition to the space state S and the transition matrix P, one needs to specify the *initial distribution*:

$$\pi_s^{(0)} = \mathbb{P}r(X_0 = s), \forall s \in S.$$

Thus, knowing $\pi^0 = \left(\pi_s^{(0)}\right)_{s\in S}$ and P, allows to compute $\pi^t = \left(\pi_s^{(t)}\right)_{s\in S}$. Indeed:

$$\pi^{(t)} = \pi^{(t-1)}P = \pi^{(0)}P^t, \forall t \geq 1.$$

Definition 2. A state s_j is *accessible* from a state s_i if the process, starting in state s_i, has a non-zero probability of reaching state s_j. This is equivalent to the following property in the transition graph G: there is an (oriented) path from s_i to s_j in G. The Markov chain is said to be *irreducible* if all its states are accessible to one another. Equivalently, G has a single strong connected component.

- A state s is *periodic* with period d if d is the smallest integer s.t. $\mathbb{P}r(X_k = s \mid X_0 = s)$ for all k that are not multiple of d. In case of $d = 1$, the state is said to be *aperiodic*.
- A state s is said to be *transient* if, the process starts in state s and there is non-zero probability that it will never return to s. A state s that is not transient, is said to be *recurrent*.
- Let s be a recurrent state and T_s be the time when the process return to s for the first time. If the expected value of T_s, given that the process starts in state s, is finite, then state s is said to be *positive recurrent*.

Now, we introduce the definition of ergodic Markov chains. This property is fundamental in the rest of this paper.

Definition 3. A state s is said to be *ergodic* if it is aperiodic and positive recurrent. In other words, a state s is ergodic if it is recurrent, has a period of 1 and it has finite mean recurrence time. If all states in an irreducible Markov chain are ergodic, then the chain is said to be *ergodic*.

Definition 4. A probability distribution $\pi^* = (\pi_s^*)_{s\in S}$ satisfying $\pi^* P = \pi^*$, *i.e.*, $\pi_{s_j}^* = \sum_{s_i\in S} \pi_{s_i}^* p_{i,j}$ for all $s_j \in S$, is called a *stationary distribution* for the Markov chain $(X_t)_{t\geq 0}$.

Then, we have the following important theorem [21]:

Download English Version:

https://daneshyari.com/en/article/9952279

Download Persian Version:

https://daneshyari.com/article/9952279

Daneshyari.com