Engineering Applications of Artificial Intelligence 60 (2017) 57–70

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

PbMMD: A novel policy based multi-process malware detection

Seyyed Mojtaba Bidoki, Saeed Jalili*, Asghar Tajoddin

Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO

Keywords:
Malware detection
Behavior-based detection
Multi-process malware
Reinforcement Learning

ABSTRACT

Contemporary malware makes wide use of techniques to evade popular detection approaches. Behavior-based detection is the most powerful approach to malware detection. This approach is based on system call sequences to model a malicious behavior. A recently immersed malware to defeat behavior-based detection approach is Multi-process malware. This malware is the consequence of multiple processes cooperating to fulfill a malicious task each of which performing a partition of main task and none of them shows an identifiable malicious behavior. In this paper, we have presented a new method called PbMMD for detecting Multi-process malware. In this method, we attempt to inspect the whole processes running on the system and discover collaborative processes by finding processes running along a common execution policy. Beforehand we have learned different execution policy by employing reinforcement algorithm. Finally we decide against a Multi-process malicious behavior by analyzing the cumulative behavior of identified collaborative processes.

1. Introduction

Malware is any software designed to carry out abusive activities like unauthorized access to system resources, disruption of operation and data and so on (Bayer et al., 2006). Every year millions of malware are produced but a lot of them are clone. So they are modified versions of known malware using techniques to evade existing detection approaches. There are two popular approaches to malware detection called signature-based and behavior-based detection (Idika and Mathur, 2007). In signature-based detection a binary code is analyzed based on its byte code structure without being executed. This approach suffers from two major drawbacks. The first one is its inability to detect unknown malware. Therefore a significant endeavor needs to be devoted to keeping update the signature repository of detection engines. These new signatures are caught by monitoring internet traffics or using honeypots. The second one is its weakness against evasion mechanisms because of its essence of inspecting sole syntactical aspect of binaries. Each procedure changing the syntactical structure with maintaining semantics can be a potential threat to this approach. There are numerous evasion techniques to signature-based detection approach like using an encryption and decryption routine in the malware (Runwal et al., 2012) and metamorphism technique (Konstantinou and Wolthusen, 2008). ROP[1] is also a technique for evading signature based detection. ROP is a technique by which an arbitrary behavior is induced in a program whose control flow has been diverted ((Roemer et al., 2012; Microsoft Security Intelligence Report – 2016)).

Another popular approach to malware detection is behavior-based detection that inspects semantic and behavioral characteristics of programs. In this way program under examination is executed in an isolated controlled system like sandbox and then its behavioral specifications like sequence of system calls are extracted for analysis (Jacob et al., 2008; Beaucamps and Marion, 2009; Bose et al., 2008; Ellis et al., 2004). Behavior-based approach often makes use of anomaly strategy for distinguishing between malware and benign programs (Chandola et al., 2009; Ellis et al., 2004). In anomaly strategy, two consecutive phases, Learning (training) phase and detection (test) phase are considered. In the learning phase by using learning techniques a set of programs (most are benign) are analyzed based on their behavioral features to extract a normal behavior (Park et al., 2010; Newsome, Song, 2005; Agrawal and Horgan, 1990). In the detection phase, the deviation amount of the program under inspection from the normal behavior forms the basis for detection. Since anomaly detection suffers from high false positive rate, specification based detection (Sekar et al., 2002) is devised to mitigate this shortcoming. In this approach the system is inspected in order to approve accurate security policies and learn all normal and acceptable behaviors according to the system rules that are applied to the system. Several major techniques have also been presented by malware writers to evade behavior based detection. An evasive approach for behavior based detection is to obfuscate system calls in order to conceal the meaningful relationship between system calls for example reordering system calls

* Corresponding author.
E-mail addresses: s.bidoki@modares.ac.ir (S.M. Bidoki), s.jalili@modares.ac.ir (S. Jalili), a.tajoddin@modares.ac.ir (A. Tajoddin).

[1] Return Oriented Programming.

http://dx.doi.org/10.1016/j.engappai.2016.12.008
Received 18 December 2015; Received in revised form 14 August 2016; Accepted 5 December 2016
0952-1976/ © 2016 Published by Elsevier Ltd.

or obfuscating the argument of system calls (Vinod et al., 2009). Lu, et al (Lu et al., 2013) presented a method called ISA[2] to defeat this evasion technique. This technique uses sequence alignment to recognize system calls obfuscations. Naval et al (Naval et al., 2015) proposed an Evasion-proof solution that is not vulnerable to system call injection attacks.

Multi-process malware is another family of malware designed for evading behavior based detection (Fan et al., 2015; Ji et al., 2014; Ma et al., 2012; Ramilli et al., 2011). This kind of malware profits from the defect of behavior-based detection approach that it focuses on a single process (and its children) for probing a suspicious behavior. For creating a Multi-process malware, a malicious task is broken into several different parts each of which is not malicious by itself (not trigger antivirus' alarm) and each part distributed into a separate process so that the processes have no parent/child relationships. Then these processes are executed such that the parts are executed in the right order of time. In the other hand the malicious task is carried out and the same cumulative effect is made as the single-process malware could make. For creating Multi-process malware an important issue in practice is how to make multiple processes execute in the right order of time. In other words how to coordinate the cooperating processes such that the main task with the real cumulative effect is performed. Therefore Multi-process malware is categorized into two different modes based on coordination method (Fan et al., 2015): 1) master/salve mode: in this mode a separate process is considered for the sole objective of coordinating collaborative processes. The processes constituting malicious task are referred to as slave processes and the coordinating process as master process. Slave processes just communicate with master process to get informed of when and how to be executed. 2) Relay race mode: in this mode the processes themselves communicate with each other with inter-process communication mechanisms in order to send required messages for coordination. In other words there is no independent separate process in charge of coordinating other processes. Multi-stage malware (Ramilli and Bishop, 2010) is another kind of evasive malware. In this way a malware is broken into several components each of which is embedded in a new file so that none of the files are identified as a malware. These files as well as a new data object called main actor is loaded on the system. Main actor is responsible for finding the files, integrating and executing them.

Till now only one approach for detecting Multi-process malware has been presented by Fan et al. (2015). This approach is specialized for analyzing a particular category of Multi-process malware called privacy-theft Multi-process malware that endanger data privacy. In this approach, first, privacy-theft malware is modeled by an abstract framework called privacy petri net based on petri net framework. Petri net is a powerful mathematical framework with formal syntactical and semantic definitions as well as graphical representation of concepts to describe program behavior specifications. Fan et al does not cover a Multi-process version of all malware families. This approach suffers from the flaw of needing definition for malicious behavior models by human, and if the Multi-process malware does not behave according to the predefined behavior model, the approach is frustrating. The approach is dependent on the way processes communicate that is called model-dependent. For example if a process employs covert channel for communication instead of simple IPCs, the approach is no longer effective. Note that Ji et al (Ji et al., 2016) proposed a spatial correlation based method to detect multiple process social bots.

Our proposed method, PbMMD,[3] analyzes Multi-process malware based on execution policy. First, in the learning phase, different execution policy are learned by employing Reinforcement Learning Algorithm. In addition, a single-process malware detector is learned by employing reinforcement learning algorithm as a classifier. In the detection phase, System calls caused by the process under analysis and the whole processes created during its life are extracted because any of the processes has the potential of cooperating with the process under analysis, then some collaborative processes are discovered by checking if they are pursuing a common execution policy. Then cumulative behavior of collaborative processes is attained by collecting system calls created by collaborative processes in temporal order and then its maliciousness is diagnosed by the single-process malware detector previously learned. Our method is a comprehensive and model independent framework that has no limitation on the kind of malware family. In addition it achieves more accurate results rather than the previous method.

The reminder of this paper is organized as follows: Section 2 describes the basic concepts required for presenting our proposed method. Section 3 presents PbMMD. Section 4 reports experimental results. Finally Section 5 draws some conclusions.

2. Basic concepts

2.1. Reinforcement Learning

2.1.1. Definition

Reinforcement learning (RL) algorithm is a machine learning algorithm engaged in specifying a method through which agents interact with an environment in order to maximize some total reward for achieving a specific goal (Sutton and Barto, 1998). In supervised learning the sample (input, output) pairs of the function prevailing over the environment is perceived and we strive to predict this function. Therefore in supervised learning we receive instant feedbacks as if a teacher takes us to the goal. In RL as the agent takes an action on the environment, it receives some feedbacks (reward) and the appropriate actions are those which maximize the total reward but at each state we are not aware of which action to be the correct one in achieving the goal. In RL unlike supervised learning we are not told what to do but only how well we have been doing in the past. What a RL program does is that it learns to generate an internal value for the intermediate states or actions in terms of how good they are in leading us to the goal and getting us to the real reward (Sutton and Barto, 1998).

RL algorithm is based on the Markov Decision Process (MDP). MDP is satisfied if three conditions are hold (Alpaydin, 2014): 1) Following a state transition model: having a model by which to know if we are at current state S, on taking action a to which next state S' we will be transferred. 2) Owning a reward model or function: To have a model or function by which we know when we move from state S to state S' what reward we receive in return. 3) Having a state transition probability model or function: To have a model to know when we take action a at state S how likely it is to be transferred to state S'. In deterministic environments we have $P_a(s,s')=1$.

2.1.2. Using RL for classification

RL can be employed as a classifier to distinguish between two classes. One example is anomaly behavior-based detection such as host based intrusion detection. We have a quick look at how RL is used for host based intrusion detection (Xu, 2010; Xu and Xie, 2005). In RL-based intrusion detection, there are two separate phases, model training phase and detection (test) phase, respectively. During model training phase what we have is audit data from a host computer including two classes of traces, i.e., the normal traces and attack (abnormal) traces as well as a reward function associated with the underlying markov model. During applying a RL algorithm like TD,[4]

[2] Iterative sequence alignment.

[3] Policy based Multi-process Malware Detection.

[4] Temporal difference.