Detecting Malicious Hosts in SDN through System Call Learning

Danai Chasaki
Department of Electrical and Computer Engineering
Villanova University
Email: danai.chasaki@villanova.edu

Christopher Mansour
Department of Computing and Information Science
Mercyhurst University
Email: cmansour@mercyhurst.edu

***Abstract*—Software Defined Networking (SDN) has changed the way of designing and managing networks through programmability. However, programmability also introduces security threats. In this work we address the issue of malicious hosts running malicious applications that bypass the standard SDN based detection mechanisms. The SDN security system we are proposing periodically monitors the system calls utilization of the different SDN applications installed, learns from past system behavior using machine learning classifiers, and thus accurately detects the existence of an unusual activity or a malicious application.**

I. Introduction

In the past few years, Software Defined Networking (SDN) has become an established novel networking paradigm that facilitates the development and management of computer networks. Although SDNs have aggressively evolved, the attack surface of SDN has become a trending research topic in order to detect and mitigate possible security breaches against SDN [1]. Adversaries can now target virtual hosts connected to the network. Shin et. al. showed an example of a malicious attack where the attack code can be stored in the payload of the packet and thus will be executed by the virtual node while the packet's header is being inspected by the SDN controller [2].

Additionally, counterfeit virtual hosts may exist that inject faults or malicious data that can corrupt services and threaten the integrity and trustworthiness of the information being exchanged.In this work we use a type of "side channel analysis" to detect any type of malicious activity or application that changes the processing behavior of the virtual host. We chose our side channel to be the operating system calls of the virtual host. We feed extracted system call features - from a mix of benign and malicious applications - into machine learning classifiers to train our security system to accurately detect suspicious activities that change the host's processing behavior and consequently the system call mix. The remainder of the paper is organized as follows: Section II discusses some related work. In Section III, we describe the methodology used to set up the SDN, trace the system calls, and train the different machine learning classifiers. Evaluation and results of the proof of concept implementation are also presented in the same section.

II. Related Work

Lee, et al. [1], tested three attack scenarios that leverage SDN applications in order to attack the SDN network. These scenarios were tested against three known SDN controllers, which are ONOS, OpenDaylight and Floodlight. Additionally, they discussed possible defense mechanisms, however, they did not propose any detection mechanism for these attack scenarios.

Canzanese, et al. [3] presented a study where they used system call analysis to detect malware that evade traditional defenses. Their system would monitor executing processes to identify compromised hosts in production environments. Rosenberg [4] implemented an intrusion detection system based on a configurable machine learning classifier. Existing literature is primarily focused on the detection of security attacks through packet inspection. In this work we focus on the analysis of SDNs system calls using machine learning to detect various types of malicious activities.

III. Methodology

In order to emulate an SDN virtual environment we created a framework consisting of an Open vSwitch (OVS) and a Floodlight controller. For our setup we used six Dockers containers, one that acts as the web Server and five that make periodical requests to the Web Server. The job of the controller is to gather system call data for each application that is running on the virtual hosts (system call type and frequency), run machine learning classifiers and decide whether the application at hand is benign or not. One of the virtual host performs a buffer overflow on the web server, which manifests into a mild type of attack in terms of system call variation, while the five containers perform a distributed denial of service attack on the web server, which leads to a severe type of attack manifestation. We use the Linux utility strace to capture and monitor the system calls for the detection of malicious applications on virtual hosts, specifically those that are exploited to perform code injection attacks. The malicious application will use system calls in a different way than what is expected. This variation may be reflected with new calls, i.e. new type of calls, that have never been used before, a different order on the calls, additional number of existing function calls.

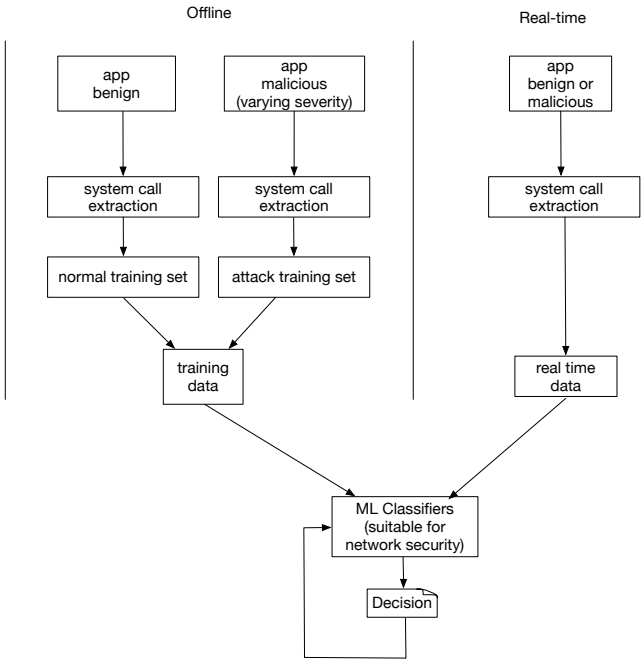

Fig. 1: System Architecture

Figure 1 presents our system's architecture. Hundreds of training data are collected in order to help the machine learning classifiers decide whether the application running on the virtual host belongs to the benign class or the malicious class (binary classification). First, there is an offline phase where benign networking applications running on our virtual hosts are analyzed. Their dominant system calls are extracted and used as normal training data sets for a handful of carefully selected machine learning classifiers. At the same time malicious applications that lead to mild, moderate and severe attacks are analyzed. Their system call mix is extracted as well, and is added to the benign training data. In real-time, the controller will periodically poll the applications while running a particular task, will extract the type and frequency of the system calls involved, and with the help of the ML classifiers determine the existence of malicious activities.

We need to note here that traditional machine learning algorithms have the intrinsic assumption that the two classes (benign vs. attack) have roughly the same number of data points. But this does not hold true for network traffic. In order to be consistent with real life we selected three training scenarios: a) 99% benign and 1% malicious traces b) 90% benign and 10% malicious traces, and c) 75% benign and 25% malicious traces. To solve the challenge posed by the class-imbalanced data sets instead of up-sampling (simply repeating the same minority class data points), we used the concept of Synthetic Minority Over Sampling Technique (SMOTE) proposed by Chawla et al [5].

The models we selected are: Random Forest, Support Vector Machine (SVM) algorithms, the Nearest Neighbors Classifier (kNN) model and the Gaussian Naive Bayes model. The first two were were selected to provide two different view points for the multi-training analysis. These two algorithms have significantly different inductive biases. The Gaussian Naive Bayes model was also selected due to the amount of features our data has. There are multiple types of system calls and the GBN model assumes that the features are independent from each other, which holds true for most system call functions. In order to evaluate the efficacy of system call learning in SDN, we used the metrics most often used by data scientists [6]: Accuracy, Precision, Recall and F1 score.

Table I shows the F1 Score obtained for the different models and the three different scenarios. An F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. As we can see, the Random Forest model got the highest F1 Score in Scenario 1, where we have very limited amount of malicious training data. This is the most common scenario in real network traffic. The k-Nearest Neighbors model got the best F1 Score on the second scenario. Finally, the Random Forest model got the highest F1 Score on the third scenario, where 25% of our training data belonged to malicious traces. The score reached almost perfect value of 0.951. These results are very encouraging, and as other network security research concludes, the Random Forest model is the ideal candidate for malicious attack detection with low number of false positives and low number of false positives as well.

TABLE I: F1 Score for the different classification models under Scenarios 1, 2 and 3[1]

Model	Scenario 1	Scenario 2	Scenario 3
GNB (F1 score)	0.627	0.659	0.776
Linear (F1 score) SVM	0.909	0.889	0.938
RBF SVM (F1 Score)	0.923	0.898	0.938
kNN (F1 score)	0.91	0.928	0.938
RF (F1 score)	0.925	0.92	0.951

REFERENCES

[1] S. Lee, C. Yoon, and S. Shin, "The smaller, the shrewder: A simple malicious application can kill an entire sdn environment," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks*, ser. SDN-NFV Security '16. New York, NY, USA: ACM, 2016, pp. 23–28. [Online]. Available: http://doi.acm.org/10.1145/2876019.2876024

[2] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (sdn)," in *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*. IEEE, 2016, pp. 1–9.

[3] R. Canzanese, S. Mancoridis, and M. Kam, "System call-based detection of malicious processes," in *2015 IEEE International Conference on Software Quality, Reliability and Security*. IEEE, 2015, pp. 119–124.

[4] I. Rosenberg, "Advanced project in computer science: System-calls based dynamic analysis intrusion detection system with configurable machine-learning classifier," The Open University of Israel, Tech. Rep.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622407.1622416

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data," *Expert Syst. Appl.*, vol. 73, no. C, pp. 220–239, May 2017. [Online]. Available: https://doi.org/10.1016/j.eswa.2016.12.035

[1]Scenario 1: 1% malicious training data, Scenario 2: 10% malicious training data, Scenario 3: 25% malicious training data