PhD Forum: Deep Learning-based Real-Time Malware Detection with Multi-Stage Analysis

Xiaoyong Yuan
Large-scale Intelligent Systems Laboratory, University of Florida

I. Introduction

Protecting computer systems is a critical and ongoing problem, given that real-time malware detection is hard. The state-of the-art for defense cannot keep pace with the increasing level of sophistication of malware. The industry, for instance, relies heavily on anti-virus technology for threat [1], [2], [3], which is effective for malware with known signatures, but not sustainable given the massive amount of malware samples released daily, as well as and its inefficacy in dealing with zero-day and polymorphic/metamorphic malware (practical detection rates range from 25% to 50%) [4].

Behavior-based approaches attempt to identify malware behaviors using instruction sequences [5], [6], computation trace logic [7] and system (or API) call sequences [8], [9]. These solutions have been mostly based on conventional machine learning (ML) models with hand-craft features, such as K-nearest neighbor, SVM, and decision tree algorithm. However, current solutions based on ML suffer from high false-positive rates, mainly because of (i) the complexity and diversity of current software and malware, which are hard to capture during the learning phase of the algorithms, (ii) sub-optimal feature extraction, and (iii) limited/out-dated dataset.

Since malware has been continuously evolving, existing protection mechanisms do not cope well with the increased sophistication and complexity of these attacks, especially those performed by advanced persistent threats (APT), which are multi-module, stealthy, and target-focused.

Furthermore, malware campaigns are not homogeneous—malware sophistication varies depending on the target, the type of service exploited as part of the attack (e.g., Internet Banking, relationship sites), the attack spreading source (e.g., phishing, drive-by downloads), and the **location of the target**.

The accuracy of malware classification depends on gaining sufficient context information and extracting meaningful abstraction of behaviors. In problems about detecting malicious behavior based on sequence of system calls, longer sequences likely contain more information. However, classical ML-based detectors (i.e., Random Forest, Naïve Bayes) often use short windows of system calls during the decision process and may not be able to extract enough features for accurate detection in a long term window.

Thus, the main drawback of such approaches is to accomplish accurate detection, since it is difficult to analyze complex and longer sequences of malicious behaviors with limited window sizes, especially when malicious and benign behaviors are interposed.

In contrast, Deep Learning models are capable of analyzing longer sequences of system calls and making better decisions through higher level information extraction and semantic knowledge learning. However, Deep Learning requires more computation time to estimate the probability of detection when the model needs to be retrained incrementally, a common requirement for malware detection when new variants and samples are frequently added to the training set.

The trade-off is challenging: fast and not-so-accurate (classical ML methods) versus time-consuming and accurate detection (emerging Deep Learning methods). **Our paper is to leverage the best of the two worlds with Spectrum, a practical multi-stage malware-detection system operating in collaboration with the operating system (OS)**.

The main idea is as follows (Figure 1). All software in the system start running in the standard OS environment and is continuously monitored through a behavioral detector based on classical ML algorithms, which provides fast detection decision. If a piece of software receives a borderline classification (i.e., reaches a threshold set by the system administrator), it is moved to the **uncertain stage**. In this stage the software will experience probabilistic and random perturbation strategies, whose severity will depend on whether the software is white-listed or not by the organization.

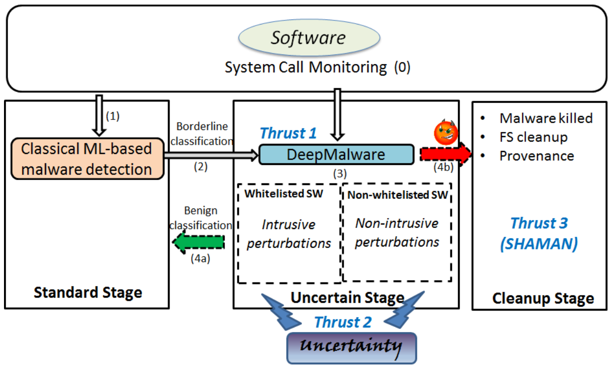

Fig. 1. Overview of DeepMalware system

The goal of the perturbation strategies is to thwart the actions of potentially stealthy malware or compromised benign software while our novel Deep Learning-component, DeepMalware, is acting. If DeepMalware finds the software benign, it is placed back in the **standard stage**, where it is

978-1-5090-6517-2/17/$31.00 ©2017 IEEE

continuously monitored. Otherwise, the software is placed in a third stage where it is killed and removed from the file system. At this stage, Spectrum also provides forensics information for system administrators, such as files tainted by malware, network connections established, etc. Newly found malware is added to the training set and all learning modules are retrained. Retraining is complex and involves designing a concept-drift module that updates the training set by adding newly relevant samples and removing newly irrelevant examples.

II. Preliminary Results

We conduct our experiments on two datasets Ubuntu 14.04 and Windows 7. In Linux dataset, We collect 100 malware from Virustotal and 400 benign applications. We run experiments in 20 Ubuntu 14.04 (32 bit) virtual machines and generate 100,000 samples based on collected malware and applications.

We designed and implemented the DeepMalware system with two-stream deep learning models (Figure 2) and multi-stage malware-detection system.

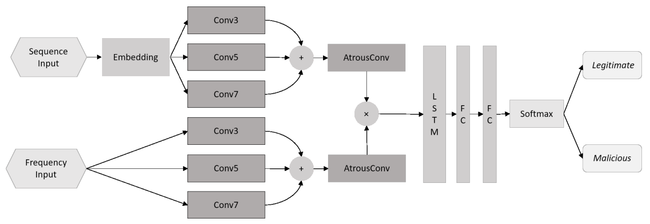

Fig. 2. Illustration of our Two-Stream model. The model consists of three main components, namely N-bag embedding, multi-scale spatial modeling and temporal modeling. N-gram embedding is only applied to the sequence input. Multi-scale spatial modeling is done with an inception-like architecture with 1×3, 1×5, 1×7 convolution respectively, where all the outputs are concatenated to extract local and global information. Two atrous convolutional layers are followed to enlarge the receptive span. Temporal modeling is conducted with LSTM model. With the combination of the features of two streams, the LSTM layer can have a global context information. We get the final prediction after two fully connected layers and a softmax activation output.

DeepMalware solved the problems of low accuracy, high model complexity and low-quality dataset in conventional machine learning based malware detection. The filter-reconstruction module allowed analyzing long sequences of system calls with large sliding windows and significantly reduced the overhead and model complexity while preserving adequate context information. The n-gram word embedding scheme precisely extracted an underlying relationship between 2-tuple system calls and facilitated the deep pipeline of CNN and LSTM layers.

In our preliminary results, Deep learning model with the 2-tuple input performed best and achieved a 94.83% accuracy and a 94.66% F1 score in Linux dataset. In term of accuracy and speed, we compare our deep learning model (dl) with conventional machine learning algorithms: random forest(rf), isolation forest(if), adb(AdaBoosting), and eXtreme Gradient Boosting(xgb) (Table I). Deep learning model shows the advantages in the accuracy, but it performs much slower than the conventional machine learning algorithms. In this regard, we propose our multi-stage malware-detection system.

TABLE I
Comparison between deep learning and conventional models

algorithm	avg time (s)	accuracy	precision	recall	f1	auc
dl	0.2921	94.36%	99.14%	89.73%	94.20%	94.46%
rf	0.0049	80.04%	75.22%	90.77%	82.26%	79.82%
if	0.0257	48.14%	49.56%	94.41%	65.00%	47.20%
adb	0.0105	84.48%	82.50%	88.30%	85.30%	84.41%
xgb	0.0102	86.21%	85.37%	88.04%	86.68%	86.17%

There still remain several opening challenges in this paper: 1) Advanced Persistent Threat (APT) takes much longer time to be processed and detected. This will cost overhead in the system for a long term. Our method mitigates its effect by compressing the system call traces. Can we provide there any other way to reduce the overhead further? 2) We are trying to reduce the detecting time by multi-stage system. But how to make sure the required measures have been taken before threat occurs?

Acknowledgment

The work presented in this paper is supported in part by National Science Foundation (grants ACI 1245880, ACI 1229576, CCF-1128805) and Florida Cyber Security grants. We'd like to thank Ruimin Sun, Qile Zhu, Aokun Chen's great contribution on this project. Especially, Ruimin Sun did great work in Linux Kernel and Window SSDT hooking. We'd like to thank Xiaolin Li, Andre Gregio, and Daniela Olivera who provided expertise and insight in this project.

References

[1] S. Kumar and E. H. Spafford, "An application of pattern matching in intrusion detection," 1994.

[2] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based intrusion detection approach," *IEEE Trans. Softw. Eng.*, vol. 21, no. 3, pp. 181–199, Mar. 1995.

[3] G. Vigna and R. A. Kemmerer, "NetSTAT: A Network-Based Intrusion Detection Approach," ser. ACSAC '98, 1998.

[4] "Bromium end point protection https://www.bromium.com/." [Online]. Available: https://www.bromium.com/

[5] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantics-aware malware detection," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (Oakland 2005)*, Oakland, CA, USA, 2005.

[6] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC-FSE '07, 2007, pp. 5–14.

[7] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Detecting malicious code by model checking," *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 3548, pp. 174–187, 2005.

[8] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09, 2009, pp. 351–366.

[9] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*. ACM, 2012, pp. 122–132.