Poster: An Early Detection of Android Malware Using System Calls based Machine Learning Model

Xinrun Zhang[1], Akshay Mathur[2], Lei Zhao[1], Safia Rahmat[2], Quamar Niyaz[1], Ahmad Javaid[2], Xiaoli Yang[3]
[1]ECE Department, Purdue University Northwest, Hammond, IN 46323, USA
[2]EECS Department, University of Toledo, Toledo, OH 46306, USA
[3]Computer Science Department, Fairfield University, Fairfield, CT 06824, USA

Abstract

Several host intrusion detection systems (HIDSs) based on system call analysis have been proposed in the past to detect intrusions and malware using relevant datasets. Machine learning (ML) techniques have been applied on those datasets to improve the performances of HIDSs. However, the emphasis given on their real-world deployment is limited. To address this issue, we propose a framework for system call processing for benign and malware Android apps with an ability of early detection of malware. We extracted and analyzed system call traces for benign and malware apps, and processed their system call traces with N-gram and TF-IDF models. Six ML algorithms – Decision Trees, Random Forest, K-Nearest Neighbors, Naive Bayes, Support Vector Machines, and Multi-layer Perceptron – were trained for the malware detection system. The experimental results demonstrate that our Android malware detection system (AMDS), using traces of 3000 system calls, is capable of early detection with an average accuracy of 99.3%. We also implemented an Android app based on a client-server architecture for the proposed AMDS to demonstrate its deployment for malware detection in real-time.

Acknowledgements

This work has been published in 2022 International Conference on Availability, Reliability and Security (ARES' 22) workshop [1].

References

1. Xinrun Zhang, Akshay Mathur, Lei Zhao, Safia Rahmat, Quamar Niyaz, Ahmad Javaid, and Xiaoli Yang. 2022. An Early Detection of Android Malware Using System Calls based Machine Learning Model. In Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES' 22). https://doi.org/10.1145/3538969.3544413

Poster: An Early Detection of Android Malware Using System Calls based Machine Learning Model

Xinrun Zhang[1], Akshay Mathur[2], Lei Zhao[1], Safia Rahmat[2], Quamar Niyaz[1], Ahmad Javaid[2], Xiaoli Yang[3]
[1]ECE Department, Purdue University Northwest, Hammond, IN 46323, USA
[2]EECS Department, The University of Toledo, Toledo, OH 46306, USA
[3]Computer Science Department, Fairfield University, Fairfield, CT 06824, USA

1. Introduction

- A large user base has made Android devices target for various malware attacks.
- Malware detection systems (MDSs) based on system call analysis have been found effective to detect malware.

Problem: Growing complexity of apps creates constraints for these MDSs to process a large number of system call traces in real-time.

Solution: Create an MDS with **early detection ability** to determine if an app is benign or malicious by analyzing a few system calls during the initial app execution.

2. Contribution

- Proposed a system call based MDS using an early detection technique.
- Built a machine learning (ML) based malware detection model through detailed analysis of system call traces from several Android malware and benign apps and feature extraction techniques.
- Collected and analyzed system call traces of 125 malware and 300 benign apps.
- Developed an Android app based on client-server architecture for deploying the MDS in real-time.

3. Methodology

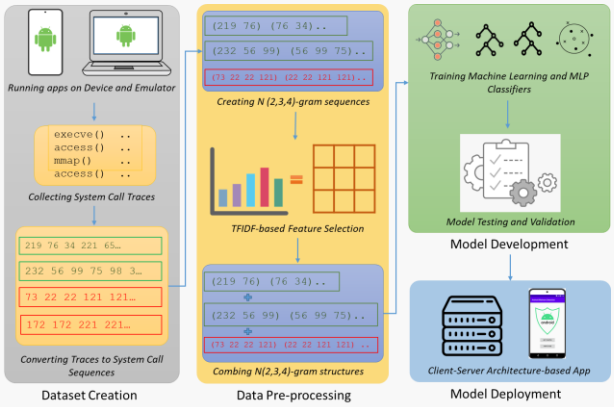

Fig.1 : MDS Development Workflow

Android apps selection

- **Malware apps**: 125 apps from Android Malware Dataset (AMD) repository.
- **Benign apps**: 300 apps with high rating downloaded from Google Play Store.

Tools used

- **Genymotion** for running apps in a virtual device.
- **Monkeyrunner** for automating apps execution using Python scripts.
- **Strace** for monitoring system calls invoked the execution of apps.
- **Scikit-Learn** library for ML model development.

Data collection and pre-processing

- Installed and executed apps in Genymotion using Python script.
- Simulated user events in each app for a minute using Monkey.
- Collected 2001 malware and 4580 benign system call traces during app execution.
- Created 2,3,4 N-gram sequences of system calls, applied Term Frequency–Inverse Document Frequency (TF-IDF), and merged N-gram sequences.

ML Model Development

- **ML algorithms used**: Decision Tree (DT), K-Nearest Neighbors (KNN), Multi Layer Perceptron (MLP), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machines (SVM).
- **Performance metrics**: Receiver operating characteristics (ROC) curve, Accuracy, and Weighted F1-score.

4. System call analysis

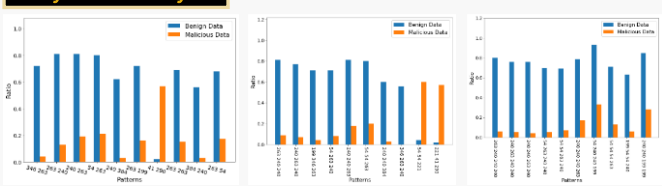

Fig.2: Top-10 **common** 2, 3, and 4-gram sequences in benign and malware traces

Key observations in system call analysis

- Analyzed top-10 frequent N-gram sequences and top-10 **common** N-gram sequences for 2, 3, and 4-gram sequences.
- Top-10 frequent N-gram sequence patterns for benign and malware traces were different. Both types of traces had some unique frequent sequences.
- Top-10 2, 3, and 4-gram **common** sequences had different distribution in benign and malware traces shown in the Fig. 2.

5. Results for ML models

- Combined 2-gram, 3-gram and 4-gram sequences for different number of initial system calls, referred to as **selection window**, from apps system call traces.
- Trained ML models for **six** classification algorithms: DT, KNN, MLP, NB, RF, SVM.
- Used ROC curves to illustrate the performance of each classifier. MLP based models have highest area under ROC curves for all selection windows shown in Fig. 3. Table 1 shows performance of MLP model.

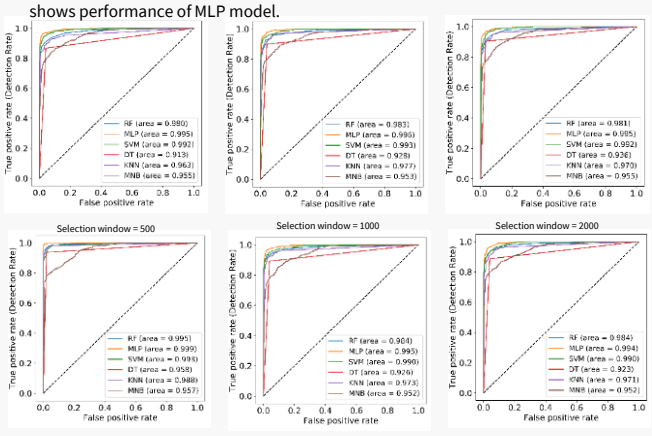

Fig.3: ROC curves for six ML classifiers models with different system call selection window

Table 1: Performance of the MLP model on test data for each selection window

Selection Window	Accuracy(%)	Precision (%)		Recall (%)		F1-Score		Weighted F1
		Malware	*Benign*	*Malware*	*Benign*	*Malware*	*Benign*	
500	97.59	97.24	97.97	95.27	98.83	0.9625	0.9840	0.9775
1000	98.29	98.31	99.2	98.14	99.27	0.9822	0.9923	0.9893
2000	99.06	98.47	99.12	97.97	99.34	0.9822	0.9923	0.9893
3000	99.3	98.99	99.63	**99.16**	99.56	**0.9907**	**0.9960**	**0.9944**
4000	99.37	99.15	99.42	98.65	99.63	0.9891	0.9952	0.9934
5000	99.32	**99.32**	**99.42**	98.65	**99.71**	0.9902	0.9956	0.9939
All traces	**99.34**	98.82	99.41	98.65	99.49	0.9873	0.9945	0.9923

6. ML Model Deployment

- Developed a client-server architecture-based Android app for deploying the MDS.
- Client runs an app and transfers the pre-processed 3000 system calls to server.
- Server runs MLP based ML model to classify the app and sends results to client.

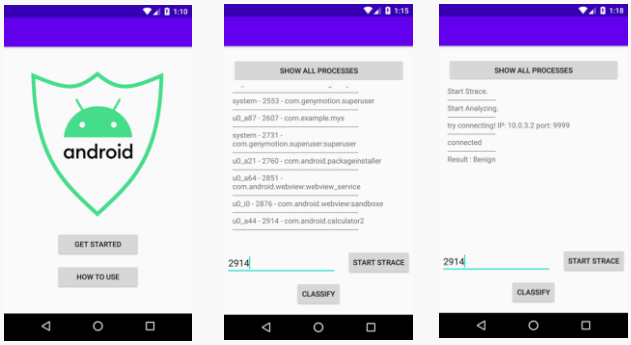

Fig.4: Android app interface for client-server based MDS

7. Conclusion

- We implemented an **early Android malware detection technique** using initial system call analysis of a running apps.
- MLP classifier performed best (accuracy 99.3%, weighted f1-score 0.9944) for a selection window of 3000 system calls.
- We are integrating this work with other static and dynamic analysis techniques to develop a robust Android malware detection engine.