Naval Information Warfare Center PACIFIC

TECHNICAL DOCUMENT 3390
July 2019

Aggregated Machine Learning on Indicators of Compromise

John M. San Miguel
Megan E.M. Kline
Roger A. Hallman
Johnny Phan
Scott M. Slayback
Christopher M. Weeden
Jose V. Romero-Mariona

Distribution Statement A: Approved for public release; distribution is unlimited.

NIWC Pacific
San Diego, CA 92152-5001

This page intentionally blank.

TECHNICAL DOCUMENT 3390
July 2019

Aggregated Machine Learning on Indicators of Compromise

John M. San Miguel
Megan E.M. Kline
Roger A. Hallman
Johnny Phan
Scott M. Slayback
Christopher M. Weeden
Jose V. Romero-Mariona

Distribution Statement A: Approved for public release; distribution is unlimited.

Administrative Notes:
This document was approved through the Release of Scientific and Technical Information (RSTI) process in June 2018 and formally published in the Defense Technical Information Center (DTIC) in July 2019.

This document's content represents work performed under Space and Naval Warfare Systems Center Pacific (SSC Pacific). SSC Pacific formally changed its name to Naval Information Warfare Center Pacific (NIWC Pacific) in February 2019

NIWC Pacific
San Diego, CA 92152-5001

NIWC Pacific
San Diego, California 92152-5001

M. K. Yokoyama, CAPT, USN
Commanding Officer

W. R. Bonwit
Executive Director

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Cyber / Science & Technology Branch (Code 58230) and Advanced Electromagnetics Technology Branch (Code 58230) of the Cybersecurity Engineering Division (Code 58220), Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA. The Naval Innovative Science and Engineering (NISE) Program at SSC Pacific funded this Applied Research project.

Released by
Jose Romero-Mariona, Head
Cyber / Science & Technology

Under authority of
Jara D. Tripiano, Head
Cybersecurity Engineering

This is a work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction.

The citation of trade names and names of manufacturers is not to be construed as official government endorsement or approval of commercial products or services referenced in this report.

MATLAB® is a registered trademark of The MathWorks, Inc.

EXECUTIVE SUMMARY

The increasing ubiquity of mobile computing technology has lead to new trends in many different sectors. “Bring Your Own Device” is one such growing trend in the workplace, because it allows enterprise organizations to benefit from the power of distributed computing and communications equipment that their employees have already purchased. Unfortunately, the integration of a diverse set of mobile devices (e.g., smart phones, tablets, etc.) presents enterprise systems with new challenges, including new attack vectors for malware. Malware mitigation for mobile technology is a long-standing problem for which there is not yet a good solution. In this paper, we focus on identifying malicious applications, and verifying the absence of malicious or vulnerable code in applications that the enterprises and their users seek to utilize. Our analysis toolbox includes static analysis and permissions risk scoring, pre-installation vetting techniques designed to insure that malware is never installed in devices on an enterprise network. However, dynamic code-loading techniques and changing security requirements mean that apps which previously passed the verification process, and have been installed on devices, may no longer meet security standards, and may be malicious. To identify these apps, and prevent future installation of them, we propose a crowd-sourced behavioral analysis technique, using machine learning to identify malicious activity through anomalies in system calls, network behavior, and power consumption. These techniques apply effectively to single user devices over time, and to individual devices within an enterprise network.

This page is intentionally blank.

CONTENTS

EXECUTIVE SUMMARY ..... **v**
1. INTRODUCTION **1**
1.1 CONTRIBUTION 1
1.2 BACKGROUND 1
1.2.1 Crowd-sourced Behavioral Analysis 1
1.2.2 Related Work 2
2. MOBILE TECHNOLOGY IN THE CONTEXT OF THE NAVY **5**
2.1 MOBILE ECOSYSTEM SECURITY GAPS 5
2.2 HOW THE NAVY IS DOING MOBILE 6
2.2.1 How the Navy is doing mobile security 6
3. THE MAVERIC APPROACH TO DYNAMIC ANALYSIS FOR MOBILE (ANDROID) APPLICATION SECURITY **9**
3.1 FEATURE SETS 10
3.1.1 Rationale for Collecting Power Consumption 10
3.1.2 Rationale for Collecting Network Activity 10
3.1.3 Rationale for Collecting Sequences of System Calls 10
3.2 DATA ANALYSIS 11
4. EXECUTION PLAN **12**
4.1 POWER CONSUMPTION 13
4.2 NETWORK ACTIVITY 13
4.3 SEQUENCE OF SYSTEM CALLS 13
4.4 APPLICATION SET 14
4.5 MACHINE LEARNING METHODOLOGY 14
5. CONCLUSION AND FUTUREWORK **18**
REFERENCES **20**

Figures

1. MAVeRiC's overall architecture makes use of an advanced static analysis capability that utilizes the Artemis tool to verify a lack of malice in Android applications. Crowd-sourced dynamic analysis monitors applications to ensure that malice is not present during application execution. Dell and Dell Precision are trademarks of Dell Inc. or its subsidiaries. Intel is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries. 3
2. MAVeRiC's approach to dynamic analysis is as follows: Known good and bad applica- tions are monitored for power consumption, network activity, and system calls. Both supervised and unsupervised machine learning techniques are utilized for detecting IOCs. 9

1. INTRODUCTION

Mobile technology has become ubiquitous in society, leading to new trends in many different sectors. “Bring Your Own Device” (BYOD) [9] is a trend that has entered many workplaces to accommodate employees’ comfort and familiarity with their personal devices. The benefits of BYOD policies include allowing companies to save money by not having to make information technology purchases and enabling a distributed computing and communications network of employees’ equipment. Estimates in 2011 suggested that nearly 75% of employers allowed employees to connect their personal devices to enterprise networks [28], and this trend has only increased since then. Indeed, the BYOD phenomena can be found in diverse sectors such as business [10], education [22], and healthcare [18]. Faced with a younger generation of workers who have always had mobile devices, government bodies at various levels within the United States are exploring the adoption of BYOD policies [23]. This phenomena has even become an issue for military organizations, where personal devices may interact with critical cyber-physical systems as well as environments that contain extremely sensitive information [7, 21].

In light of this new reality, military and other government organizations must determine ways to keep malicious applications on personal devices from infecting corporate networks. To this end, we propose Mobile Application Vetting and Risk-Estimation Capability (MAVeRiC), a program which makes use of both static and dynamic analysis to vet Android[1®] applications. Specifically, MAVeRiC offers the ability to vet Android applications for the absence of malice both pre- and post-installation. This post-installation vetting is accomplished by comparing data from running applications between users on enterprise networks. MAVeRiC’s overall architecture can be seen in Figure 1.

1.1 CONTRIBUTION

Our main contribution in this paper is the description of an approach for verifying the absence of malice in Android applications that utilizes a conglomeration of machine learning techniques on crowd-sourced behavioral data. We also provide background information on how enterprise networks which host enclaves with particularly sensitive information and critical systems handle the BYOD phenomenon, which informs our approach in MAVeRiC.

1.2 BACKGROUND

Static (or code) analysis provides an analysis of an application without actually executing it. One such analysis technique is to create “feature vectors” that characterize a given application’s characteristic actions (e.g., permissions requests). Benign applications within each category that have similar functions are expected to have similar permissions requests, while malicious ones deviate; the extent of deviation is measured and used for risk assessment. Almost all static analysis for risk assessment of Android applications use permission calls as their main criteria. One weakness of static analysis techniques is a vulnerability to code obfuscation. Dynamic (or behavioral) analysis is not vulnerable to code obfuscation or other evasion techniques that can get past a static analysis regimen because it is able to observe malicious behavior on an execution path [29]. Our prior work [11] provides a brief survey of static and dynamic analysis tools for Android Applications.

[1]The Android name and Android Robot logo are property of Google LLC. The Android Robot logo is licensed under the terms of the Creative Commons Attribution 2.5 license.

1.2.1 Crowd-sourced Behavioral Analysis

The Crowd-Sourced Behavioral Analysis (CSBA) approach outlined in this paper is part of a larger framework of technology and policy that the MAVeRiC team is developing. MAVeRiC employs both a pre-installation vetting procedure as well as post-installation dynamic analysis. The pre-installation vetting includes triage, data flow analysis, and permissions risk analysis, which all feed into an overall risk score.

The triage and data flow analysis portion of the vetting process is built around Artemis (which utilizes DroidSafe's static analysis capabilities) [17], the best-of-breed solution to the DARPA Automated Program Analysis for Cybersecurity (APAC) program[2], provided by Raytheon BBN[3]. Triage does a quick comparison of short sequences of machine language instructions against sequence lists from Android Packages (apks) that are known to be malicious. Data Flow analysis lifts apk binaries to an intermediate representation (IR) called Jimple [27], and the framework generates a listing of the possible execution paths of the program. This listing can be shown to analysts in the form of a control-flow-graph (CFG), or a class-call-graph (CCG) with the ability to query results for specific data flows. Permissions risk analysis is based on the likelihood of a given application's category requesting a certain set of permissions. The overall risk score is a quantification of the individual analyses to provide analysts with an aggregate overview of an application's risk. The overarching goal is to support non-specialized IT personnel in quickly evaluating the risk involved with installing a given app on a DoD device.

The focus of this paper is to describe the Crowd-Sourced Behavioral Analysis (CSBA) approach used to develop MAVeRiC's post-installation dynamic analysis. MAVeRiC crowd-sources data and uses a machine learning approach to analyze sequences of system calls, network behavior, and power consumption data to identify malicious activities both from a single user's device over time as well as within a trusted network of users.

1.2.2 Related Work

Other related efforts have validated the approach MAVeRiC has taken. One paper suggests that there are limitations to utilizing purely static or dynamic approach to analyzing Android malware [5]. They point out that the standard of static malware could not be sufficient to protect against techniques that evade or obfuscate such analysis and are therefore inadequate and ineffective to an analyst. They propose an approach that detects Android malware by fingerprinting the behavior of system calls and incorporating machine learning to be able to associate malicious behaviors. The approach is validated using a real device and experiments on 20,000 execution traces across 2,000 applications with a 97% accuracy. Our approach to analyzing sequences of system calls will closely mirror theirs by being run on an emulator, but then validated with a real device metrics. In addition, we are incorporating other types of data for analysis, including network activity and power consumption.

---

[2] `https://www.darpa.mil/program/automated-program-analysis-for-cybersecurity`

[3] Raytheon is a registered trademark of the Raytheon Company.

An approach we leverage that is validated in another paper [6] discusses the use of power consumption as a mechanism to create an energy footprint to determine a baseline. Along with the baseline, they propose the use of energy consumption measurement from seven covert channels including type of intent, file lock, system load, volume settings, unix socket discovery, file size, and memory load.

Inspired by [4], we also make use of the crowd-sourcing paradigm. By collecting data from multiple users within a network that is semi-trusted, we have a much more robust picture of how apps are used and can get a better understanding of what typical behaviors are. We anticipate that this with support our analytics by allowing quick identification of unususal behaviors. This work expands on that of Burguera et al. by working in a larger test environment with more features. We are also appling the approach to a Navy-relevant environment with navy security concerns in mind.

The remainder of this paper is laid out as follows: Section 2 provides a very high-level overview to the way that the United States Navy is approaching the incorporation of mobile technologies and attempting to

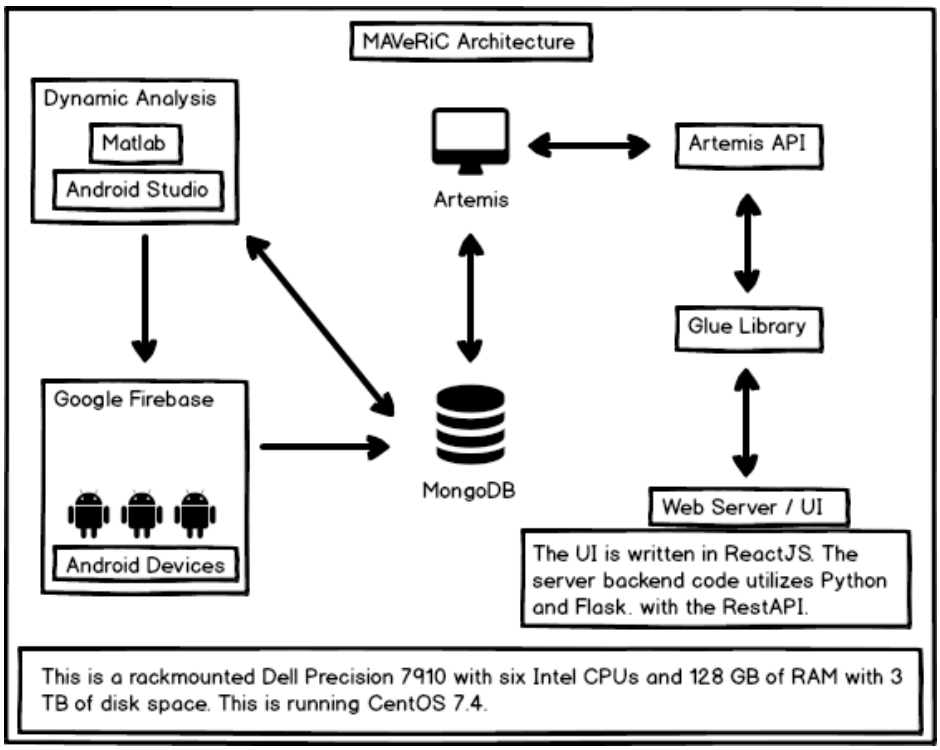

Figure 1. MAVeRiC's overall architecture makes use of an advanced static analysis capability that utilizes the Artemis tool to verify a lack of malice in Android applications. Crowd-sourced dynamic analysis monitors applications to ensure that malice is not present during application execution. Dell and Dell Precision are trademarks of Dell Inc. or its subsidiaries. Intel is a trademark of Intel Corpo- ration or its subsidiaries in the U.S. and/or other countries.

adjust to the realities of BYOD. We detail our approach to MAVeRiC dynamic analysis in Section 3 and our plan for executing the approach in Section 4. Finally, concluding remarks and directions for possible future work are given in Section 5.

This page is intentionally blank.

2. MOBILE TECHNOLOGY IN THE CONTEXT OF THE NAVY

Mobile devices are transforming the way that the navy operates. By leveraging the computing power, small form factor and many integrated sensors, we have the ability to be more responsive and interactive with our environment. There is great potential in operational use of distributed computing resources to enhance users' situational awareness, share data, build a better picture of the operating environment, and decrease out-of-pocket time. The navy has the option to leverage this computing and communications ability at a substantially reduced cost through the use of BYOD policies.

The mobile ecosystem is constructed around the use of dedicated single-purpose applications (apps), which interface with a device's onboard sensors and network communications to provide services to the device user. Since each user has different roles and needs, they will need the ability to install different apps. To meet warfighter needs, the navy can develop its own apps, and simultaneously leverage Commercial

Off-the-Shelf (COTS) apps. In either case, we need to ensure that these apps do not leak sensitive personal or mission-related information.

2.1 MOBILE ECOSYSTEM SECURITY GAPS

While the Play Store and the associated mobile ecosystem have been active for almost 9 years, security research in the field lags behind. There have been a number of incidents that demonstrate that current mobile malware detection and prevention techniques are largely ineffective.

For example, in 2017, Kaspersky labs discovered a malicious app known as SkyGoFree [2], which had been available on third-party app stores and side-loading websites since 2014. This app has several advanced features, including the ability to selectively record data from the camera and microphone, with GPS location being used as a primary selection criteria. This allows the phone to record, for example, every conversation its owner has in the company office. SkygoFree can also use assistive technology, such as screen readers, to read information from otherwise well-protected encryption applications like Whatsapp. All recorded data, user contacts, and stored personal information is exfiltrated when the app surreptitiously connects to Wi-Fi networks–even if the device is in airplane mode. As an afterthought, the app will also use the infected devices to engage in SMS and click-fraud.

Conventional wisdom – and the SkyGoFree case – suggest that the strongest protection against this sort of malicious app is to only install apps from the official stores. While this strategy may provide more protection, it doesn't provide complete protection. As an example, the ExpensiveWall malware [8, 19] – which engaged in SMS fraud, pay-per-click fraud, and data exfiltration – worked its way into the Google Play Store[4] as a variety of mobile wallpaper apps. The developers used packing techniques to obfuscate the malicious code in their APKs, a technique that successfully bypassed Play Store security measures. When the malware was discovered, researchers estimated that it had managed to infect approximately 21 million devices.

[4] https://play.google.com/store

In an even more significant case, the Facebook Messenger app, actively in use on 1.2 billion devices in 2017, was recently shown to be collecting not just user-provided information, but also SMS and call data from users' devices. This data collection may be in violation of Facebook user consent policies and a 2011 agreement between Facebook and the Federal Trade Commission. This data was stored for years on Facebook servers before being scraped and parsed by Cambridge Analytica during the 2016 election cycle [14]. The same information could just as easily have been scraped and used by adversaries seeking to construct professional and personal networks that could be used for social engineering and other intelligence gathering.

This sort of intelligence gathering was achieved by accident, when the Strava mobile fitness app published a heat map of popular running routes around the world. Researchers very quickly identified US forward military bases in the Middle East, as well as facilities in use by other militaries [12]. This app, which was completely open about its data collection and usage, had managed to reveal sensitive, mission-critical information and place users at risk by revealing common patterns of movement and behavior.

The DoD and its personnel are also deliberately targeted by malicious apps. According to the US Department of Defense website's Mobile App Gallery[5], there is an unsanctioned in-the-wild app targeting Thrift Savings Plan (TSP) participants who want to manage their retirement savings from their mobile devices. The app, "TSP Funds", which was not developed by any organization associated with the DoD or the TSP program, prompts the user to provide a username and password, enabling the developers to gain unauthorized access to the sensitive financial information of DoD personnel.

Even sanctioned apps present an attack surface that is in need of examination. DoD has released an app called Defense Finance and Accounting Service (DFAS). This app provides DoD employees with access to information about their salaries, taxes and benefits. To an adversary, this is a treasure trove of Personally Identifiable Information (PII), including personal finances and social security information.

These incidents demonstrate a need for improvement in the field of mobile security. Each app, whether intentionally or not, exposed sensitive information to malicious actors. The MAVeRiC effort seeks to identify apps that expose information improperly, and either prevent their installation, or remove them if their inappropriate behavior is discovered later.

2.2 HOW THE NAVY IS DOING MOBILE

The navy is developing policies to support the adoption of mobile devices and apps. In some areas, Android and iOS devices are being prepared by administrative staff, then issued to navy personnel. End-users may use the devices for email, telecommunications, and other business related functions. There are also pilot programs seeking a cost savings through the use of BYOD policies. These devices are granted access to business related functions and communications, but they are also integrated into users' personal lives, and apps installed on the devices reflect this. Both government-issued devices and BYOD devices are subject to Mobile Device Management (MDM) tools, which allow the organization to remotely manage device security controls, limit app installation, track device activity, and erase data from a device.

[5] https://www.defense.gov/Resources/Developer-Info/Apps-Gallery/

2.2.1 How the Navy is doing mobile security

One approach to mobile security is containerization, as demonstrated by Good Technology's Secure Enterprise Mobility Management (EMM) Suites. In principle, a containerization solution can completely partition a phone into two (or more) isolated environments, such that data in one container is not accessible to apps in another container. Even if one container were compromised by a malicious app, the other containers would be able to maintain data confidentiality. In practice, researchers have demonstrated that malicious apps in one container can gain access to underlying kernel modules and, from there, access the processes and data that should be isolated within different containers [13].

The Defense Information Security Agency (DISA) is the entity responsible for policy and practices related to mobile device and app security for the US Government. According to the DISA DOD Mobility Applications webpage[6], each app is vetted via a rigorous process including static, dynamic, and network analysis. The results of these analyses are mapped against requirements from the Mobile Application Security Requirements Guide (MAppSRG) [26], National Information Assurance Partnership (NIAP)[7], Protection Plan (PP), and the Open Web Application Security Project (OWASP)[8], among others.

DISA reports that "over 200 approved apps" are available for download in the DoD. As of December 2017, the Google Play Store has 3.5 million apps. A major goal of the MAVeRiC project is to accelerate this decision-making process, while maintaining or improving the accuracy of detection for malicious code. This improvement is vital if the DoD intends to bring the use of mobile devices and apps to its full potential.

[7] https://www.niap-ccevs.org/
[8] https://www.owasp.org/

This page is intentionally blank.

3. THE MAVERIC APPROACH TO DYNAMIC ANALYSIS FOR MOBILE (ANDROID) APPLICATION SECURITY

Dynamic Analysis for malware detection in mobile devices is a popular area of research that still has not produced a reliable malware detection framework. Many researchers have explored this topic, but each tends to focus their attention on a single Indicator of Compromise (IOC). An IOC is a measurable event that can be identified on a host or network [15] and which may indicate the presence of a compromise within that system. A single IOC does not provide sufficient confidence to reliably claim that malware is present on a mobile device. For example, a malicious app that continuously transmits recordings from the device camera and microphone will have a significant impact on device power consumption, but so will playing a game with high-resolution graphics.

The MAVeRiC framework collects data related to three different IOCs: power consumption, network behavior, and sequences of system calls. The complete feature set is analyzed using machine learning techniques to detect anomalies and classify them as benign or malicious. This is a holistic approach to detecting malicious or unintended behaviors within applications and can provide greater accuracy over models which rely on a single IOC. This paper presents an approach to finding the best machine learning methodology for detecting malicious behavior in Android applications using multiple IOCs.

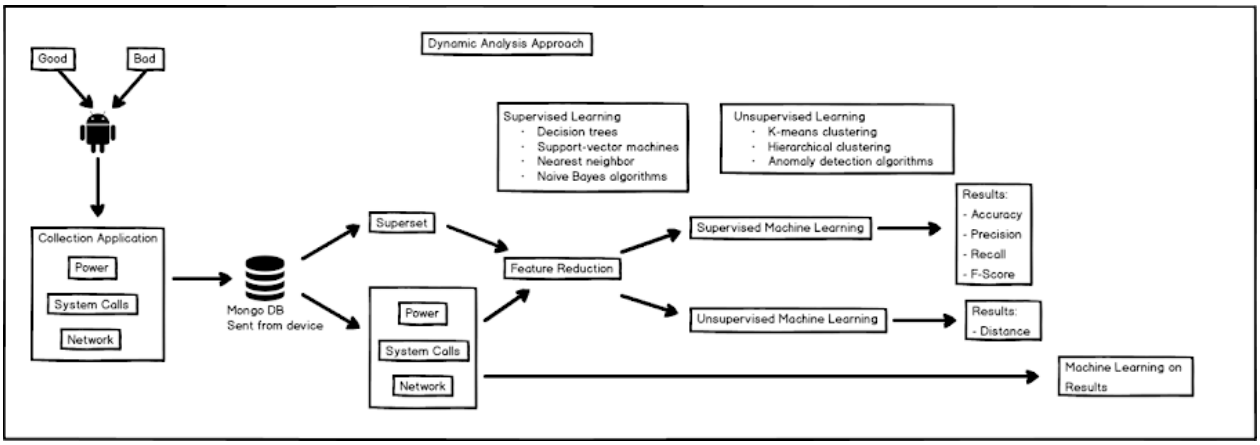

Figure 2. MAVeRiC's approach to dynamic analysis is as follows: Known good and bad applica- tions are monitored for power consumption, network activity, and system calls. Both supervised and unsupervised machine learning techniques are utilized for detecting IOCs.

3.1 FEATURE SETS

3.1.1 Rationale for Collecting Power Consumption

The power consumption of an app presents an indicator of compromise for an analyst. Power consumption varies depending on the state and activities of the apps on a device. Collecting information on power consumption allows researchers to construct baselines for expected power consumption of a device based on which apps are running at a given time. Discrepancies serve as IOCs that should be investigated for possible malice. There has been some success in using machine learning approaches to detect malicious activity on covert channels. The effort described in [6] provides a detection framework that collects power-related data using the PowerTutor[9] application and relies on regression-based and classification-based methods. The MAVeRiC capability leverages their approach towards collecting expected power consumption of mobile apps as well as analyzing features specific to power consumption.

3.1.2 Rationale for Collecting Network Activity

Network activity is an IOC that should be considered when identifying malicious behavior of mobile apps. Many of the components and programs installed on a mobile device are the same as those found on conventional computers, and so mobile devices share vulnerabilities with their larger counterparts, especially in regards to network communications. Mobile devices are nearly always communicating via network connections, whether on cellular or WIFI networks. Many of the legitimate applications on a mobile device are constantly polling the network to see if any new application information is available. MAVeRiC collects data on the state of all network communications. For each app, it is important to know the amount of data being sent, the frequency of send/receive communications, whether the app is running in the foreground or the background, etc [20]. This data is vital in understanding the normal behavior of apps–individually and collectively–and identifying when there may be malicious or unexpected activity.

[9] http://ziyang.eecs.umich.edu/projects/powertutor/

3.1.3 Rationale for Collecting Sequences of System Calls

The sequence in which system calls are made has also shown to be an important IOC for detecting malware in an Android device. System calls are how an application accesses operating system services [3]. These are underlying actions that user-level processes cannot be trusted to perform on their own, but which need to be performed in order to provide full application functionality. System calls allow these actions to be delegated to the trusted authority of the operating system kernel [1]. System calls can be organized into multiple categories [3]:

1. Process Control
2. File Management
3. Device Management
4. Information Management
5. Communication

The sequence with which system calls are called may indicate that an application is behaving maliciously. By capturing execution trace information of a given mobile application, researchers can analyze how an app uses the more than 250 system calls that are provided by the Android OS. In [5], researchers captured patterns of application behavior and constructed a fingerprint based on frequency of system calls within a given time frame. Then they used a probabilistic approach to find outliers in the expected values of these frequencies. MAVeRiC also captures system call sequences as part of the dynamic analysis to compare expected sequences over time for a given application. Anomalies in system call sequences serve as IOCs that may identify malware that is executed at random times and would not otherwise be easy to distinguish during normal operation.

3.2 DATA ANALYSIS

MAVeRiC evaluates two distinct approaches for evaluating the effectiveness of using machine learning to identify malicious behavior of an application. Both approaches use machine learning algorithms to assess the data collected from the three IOCs described above.

The first approach combines all three sets of IOC data into a single superset. The entire superset is assessed by multiple machine learning algorithms. During this phase, the data is run through feature selection algorithms to reduce the number of individual features under test. This increases the speed and efficiency of identifying malicious behavior of an application on a device. Both supervised and unsupervised algorithms are examined in the evaluation process, as described in Section 4.5.

The second approach evaluates each of the IOCs separately, then subjects the results to further analysis. Previous research has looked at all three of these IOCs individually. Those efforts have also evaluated multiple machine learning algorithms for each IOC and shown that different algorithms are most effective for the distinct IOCs. MAVeRiC attempts to recreate this previous work and use the results to populate a new data set for evaluation. One evaluation technique is to nest machine learning algorithms, where the initial results of one algorithm are then analyzed through another algorithm. The second option involves developing an algorithm where the collective results of the IOCs' machine learning algorithms are used as inputs. The outcomes of these approaches are the basis for the comparative study and evaluation of which approach performs the best.

This page is intentionally blank.

4. EXECUTION PLAN

MAVeRiC builds upon previously published work to collect data from multiple IOCs. The data is collected through separate applications and then sent to an off-device MongoDB[10] server. Then the data is analyzed using machine learning methods. A description of our data collection methods follows.

4.1 POWER CONSUMPTION

Power Consumption of a given application is not a measurement that can be analyzed using static methods. It must be monitored while the app is running on a device. All devices are not created equal, and there are many factors that contribute to the rate at which power is consumed on a given device. We are using an on-device tool named PowerTutor to collect power usage statistics. The official PowerTutor repository was last updated in April 2013, but we have forked the code and modified it to run with more current versions of Google's Android API. We have also added functionality that enables users to send collected data to an off-device server. PowerTutor collects data from a running application as well as power consumption records for each hardware component used by that app. PowerTutor models the power usage of the following hardware components: CPU, OLED/LCD, WIFI, Cellular Network, GPS, and audio. Attributing changes in these hardware component values to individual apps installed on a device will help in understanding the power usage patterns of apps within our control group.

4.2 NETWORK ACTIVITY

Capturing network activity is important for correlating network behaviors and patterns within mobile apps to characterize baseline behavior. At install, static analysis of an application may not capture the elements to detect maliciousness in the network patterns. MAVeRiC analyzes deviations in network behavior to identify malicious activity. The need for dynamic analysis of network behavior stems from weaknesses in static analysis to address apps that introduce malicious code at runtime or when updates are installed [16]. MAVeRiC leverages the Wireshark plugin, Android dump[11] to collect and aggregate both cellular and Wi-Fi network activity, then sending the data off-device to the server for analysis.

4.3 SEQUENCE OF SYSTEM CALLS

Sequences of system calls can be used to identify common app behaviors and distinguish between benign activities and potentially malicious ones. Prior to installation, the only way to know how an app will communicate with the system is by validating its binary code against app permissions, as listed in the APK manifest file. This knowledge may be incomplete, due to techniques–such as code obfuscation and custom permissions [16]–that are designed to deceive static analysis methods. App-operating system interaction is an observable trait that can be used to categorize app in terms of both core functionality and malice. Android Debug Bridge (ADB) [24], provided by the Android framework, is a tool that can communicate with an Android device. Using the ADB *Strace*

[10] `https://www.mongodb.com/`
[11] `https://www.wireshark.org/docs/man-pages/androiddump.html`

function, MAVeRiC collects the system calls an app requests during use. In order to generate a sufficient volume of data for analysis, MAVeRiC employs a tool called Monkey[25] to generate pseudo-random user activity with an application. The collected inputs and system call sequences are sent to the off-device server for further analysis.

Over time, we expect sequences of identical or similar user inputs (e.g., Monkey-generated clicks, touches, gestures...) to a benign app to produce identical or closely related system call patterns. If we malicize an app by inserting malicious functionality, we expect different system call patterns in response to the same user inputs. The behavioral differences between benign and malicious apps are used to train our machine learning systems in the classification of malicious behavior.

4.4 APPLICATION SET

Our application dataset has been constructed from a combination of sources. MAVeRiC has obtained known malicious applications from the Drebin[12] and Androzoo[13] repositories and a set of benign apps from the Google Play Store. We have also generated a control group of apps for testing by adding specific malice to certain classes of applications with desired traits. We leveraged open-source applications from the F-Droid[14] repository, for which source code is available, and inserted malicious functionality. We then re-package them into a malicized version of the original benign app. MAVeRiC is using data collected from these apps to measure the effectiveness of each of the machine learning algorithms.

4.5 MACHINE LEARNING METHODOLOGY

As discussed in the previous section we are evaluating two approaches to determine how MAVeRiC will detect malicious apps on a device. In the first approach MAVeRiC combines all of the data from the three separate IOC's into a single superset. This process entails a training period consisting of feature selection and model selection. MAVeRiC is constantly collecting data from mobile devices, and so power drain on a device is a concern and collecting all features may be expensive. MAVeRiC is looking for the features that are most relevant for making predictions. Prior to the training phase the data is run through some feature selection methods to determine which features are the most relevant. Once an optimized set of data is selected MAVeRiC inputs the data through both supervised and unsupervised machine learning models.

[12] `https://www.sec.cs.tu-bs.de/~danarp/drebin/`
[13] `https://androzoo.uni.lu/`

[14] `https://f-droid.org/`

In this approach the supervised learning model is for classification purposes. Our dataset of benign and malicious apps allows MAVeRiC to baseline the behaviors of a known malicious application. The data for both the benign and malicious apps is collected and run through several classification algorithms including decision trees, support-vector machines, nearest neighbor, and Naïve Bayes algorithms. Initially all data is evaluated using MATLAB[15]®, as it has pre-built functions for performing these tests. Each method is evaluated based on its:

1. Accuracy – $\frac{\text{number of correct predictions}}{\text{number of items in total dataset}}$.
2. Error rate – $\frac{\text{number of incorrect predictions}}{\text{number of items in the total dataset}}$.
3. Recall – $\frac{\text{number of correct positive predictions}}{\text{number total observed positives}}$.
4. Precision – $\frac{\text{number of correct positive predictions}}{\text{number of total predicted positives}}$.
5. F-score – $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$.

In the case of unsupervised learning model MAVeRiC is utilizing both clustering and anomaly detection. The unsupervised learning approach will include k-means clustering, hierarchical clustering, and anomaly detection algorithms. In the both the clustering and anomaly detection methods the center of the clusters is the value from which all points will be measured. A threshold is created, representing the distance from center point. All points outside of the threshold are identified as being potentially malicious.

In the second approach each IOC is evaluated individually. MAVeRiC analyzes each IOC, following the same steps as in the first approach. A feature selection reduction takes place on the individual IOC's, as they may differ than those chosen as a superset in the first approach. Once a set of features is selected the data is input into a series of supervised machine learning algorithms in order to train the model.

We expect MAVeRiC's dynamic analysis approach to show a reduction in both False-Positive andFalse-Negatives, when multiple IOC's are analyzed. Once all the approaches are analyzed we will evaluate each based on multiple factors, in order to make a determination on which is best. The tradeoff analysis will be based on: accuracy, speed, amount of data to collect, resources utilized on Android devices. MAVeRiC will then integrate the selected approach into the framework by deploying data collection applications on Android devices and servers to evaluate the data. MAVeRiC will also work to develop a method of introducing additional IOCs in the future. We will then integrate the selected approach into the framework by deploying data collection applications on Android devices and servers to manage the data. MAVeRiC will also work to develop a method of introducing additional IOC's in the future.

[15]MATLAB® is a registered trademark of The MathWorks, Inc.

This page is intentionally blank.

5. CONCLUSION AND FUTUREWORK

The proliferation of mobile computing platforms, as well as the much-needed applications and content to make them useful, continues to accelerate. One of the biggest challenges, as a result of this proliferation, are the potential security risks and vulnerabilities that could be found in mobile applications. In this paper, we introduced a new conceptual technology called MAVeRiC, which aims at bringing better security solutions to the Android mobile applications arena. Through MAVeRiC, we are moving one step closer to providing much more secure platforms and applications to users, while ensuring that this security does not become a road-block. MAVeRiC leverages novel approaches in crowd-sourced behavioral analysis and machine learning to take the guess work out of determining if an application is malicious or not, and furthermore, continue to monitor it after installation without hindering performance and/or user attention.

Furthermore, the paper described how the US Navy is moving forward with mobile platforms and applications, in order to bring a context to MAVeRiC as well as outline areas of potential future collaboration. From the research, to the technological, to the policy perspective, there are multiple areas that while MAVeRiC aims to support in the future, collaborations with industry and academia will prove essential to fulfill.

While the early results described are promising, next steps in MAVeRiC's development include a larger test experiment and demonstration in order to determine scalability issues and usability of features to a larger user audience. In addition, much more diverse malware suite, including both legacy and current threats, will be used.

This page is intentionally blank.

REFERENCES

1. Kernel space. Available at http://www.linfo.org/kernel_space.html (05 April, 2018), 2005.

2. Skygofree — a hollywood-style mobile spy. Available at https://www.kaspersky.com/blog/skygofree-smart-trojan/20717/ (06 April, 2018), 2018.

3. T. Bower. 1.12. system calls — operating systems study guide. Available at http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html (05 April, 2018), 2015.

4. I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 15–26. ACM, 2011.

5. G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio. Detecting android malware using sequences of system calls. In *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, pages 13–20. ACM, 2015.

6. L. Caviglione, M. Gaggero, J.-F. Lalande, W. Mazurczyk, and M. Urbański. Seeing the unseen: revealing mobile malware hidden communications via energy consumption and artificial intelligence. *IEEE Transactions on Information Forensics and Security*, 11(4):799–810, 2016.

7. D. Dasgupta, A. Roy, and D. Ghosh. Multi-user permission strategy to access sensitive information. *Information Sciences*, 423:24–49, 2018.

8. T. Fox-Brewster. Google is fighting a massive android malware outbreak — up to 21 million victims, 2017.

9. A. Ghosh, P. K. Gajar, and S. Rai. Bring your own device (byod): Security risks and mitigating strategies. *International Journal of Global Research in Computer Science (UGC Approved Journal)*, 4(4):62–70, 2013.

10. K. Giotopoulos, C. Halkiopoulos, D. Papadopoulos, and H. Antonopoulou. Adoption of bring your own device (byod) policy in marketing. In *5 th International Conference on Contemporary Marketing Issues ICCMI June 21-23, 2017 Thessaloniki, Greece*, page 342, 2017.

11. R. A. Hallman and M. Kline. Risk metrics for android (trademark) devices. Technical report, Space and Naval Warfare Systems Center Pacific San Diego United States, 2017.

12. J. Hsu. The strava heat map and the end of secrets. Available at https://www.wired.com/story/strava-heat-map-military-bases-fitness-trackers-privacy/(19 April, 2018), 2018.

13. U. Kanonov and A. Wool. Secure containers in android: the samsung knox case study. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 3–12. ACM, 2016.

14. T. B. Lee. Facebook's cambridge analytica scandal, explained. Available at https://arstechnica.com/tech-policy/2018/03/facebooks-cambridge-analytica-scandal-explained/ (19 April, 2018), 2018.

15. H.-Y. Lock and A. Kliarsky. Using ioc (indicators of compromise) in malware forensics. *SANS Institute InfoSec Reading Room*, 2013.

16. L. Onwuzurike, M. Almeida, E. Mariconti, J. Blackburn, G. Stringhini, and E. De Cristofaro. A family of droids: Analyzing behavioral model based android malware detection via static and dynamic analysis. *arXiv preprint arXiv:1803.03448*, 2018.

17. J. Perkins and M. Gordon. Droidsafe. Technical report, Massachusetts Institute of Technology Cambridge United States, 2016.

18. F. Portela, A. M. da Veiga, and M. F. Santos. Benefits of bring your own device in healthcare. In *Next-Generation Mobile and Pervasive Healthcare Solutions*, pages 32–45. IGI Global, 2018.

19. E. Root, A. Polkovnichenko, and B. Melnykov. Expensivewall: A dangerous 'packed malware on google play that will hit your wallet. Available at https://blog.checkpoint.com/2017/09/14/expensivewall-dangerous-packed-malware-google-play-will-hit-wallet/ (07 April, 2018), 2017.

20. A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici. Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, 43:1–18, 2014.

21. R. S. Shaji, V. S. Dev, and T. Brindha. A methodological review on attack and defense strategies in cyber warfare. *Wireless Networks*, pages 1–12, 2018.

22. Y. Song and S. C. Kong. Affordances and constraints of byod (bring your own device) for learning and teaching in higher education: Teachers' perspectives. *The Internet and Higher Education*, 32:39–46, 2017.

23. M. Souppaya and K. Scarfone. Users guide to telework and bring your own device (byod) security. *NIST Special Publication*, 800:114, 2016.

24. A. Studio. Android debug bridge (adb). Available at https://developer.android.com/studio/command-line/adb.html (23 April, 2018), 2018.

25. A. Studio. Ui/application exerciser monkey. Available at https://developer.android.com/studio/test/monkey.html (23 April, 2018), 2018.

26. Unified Compliance Framework, 244 Lafayette Circle, Lafayette, CA 94549. *Mobile Application Security Requirements Guide*, 2014.

27. R. Vallee-Rai and L. J. Hendren. Jimple: Simplifying java bytecode for analyses and transformations. 1998.

28. M. Viveros. The pros and cons of 'bring your own device'. Available at https://www.forbes.com/sites/ciocentral/2011/11/16/the-pros-and-cons-of-bring-your-own-device/\#2a0acb662abe (20 April, 2018), 2011.

29. L.-K. Yan and H. Yin. Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In *USENIX security symposium*, pages 569–584, 2012.

INITIAL DISTRIBUTION

84300	Library	(1)
85300	Archive/Stock	(1)
58230	J. San Miguel	(1)
58230	M. Kline	(1)
58230	R. Hallman	(1)
58230	J. Phan	(1)
58230	S. Slayback	(1)
58230	C. Weeden	(1)
58230	J. Robero-Mariona	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

This page is intentionally blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Department of Defense, Washington Headquarters Services Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*: July 2019

2. REPORT TYPE: Final

3. DATES COVERED *(From - To)*:

4. TITLE AND SUBTITLE

Aggregated Machine Learning on Indicators of Compromise

5a. CONTRACT NUMBER:

5b. GRANT NUMBER:

5c. PROGRAM ELEMENT NUMBER:

5d. PROJECT NUMBER:

5e. TASK NUMBER:

5f. WORK UNIT NUMBER:

6. AUTHORS

John M. San Miguel
Megan E.M. Kline
Roger A. Hallman
Johnny Phan
Scott M. Slayback
Christopher M. Weeden
Jose V. Romero-Mariona
NIWC Pacific

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NIWC Pacific
53560 Hull Street
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER: TD 3390

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Innovative Science and Engineering (NISE) Program (Applied Research)
NIWC Pacific
53560 Hull Street
San Diego, CA 92152–5001

10. SPONSOR/MONITOR'S ACRONYM(S): NISE

11. SPONSOR/MONITOR'S REPORT NUMBER(S):

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Statement A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

This is work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction.

14. ABSTRACT

The increasing ubiquity of mobile computing technology has lead to new trends in many different sectors. "Bring Your Own Device" is one such growing trend in the workplace, because it allows enterprise organizations to benefit from the power of distributed computing and communications equipment that their employees have already purchased. Unfortunately, the integration of a diverse set of mobile devices (e.g., smart phones, tablets, etc.) presents enterprise systems with new challenges, including new attack vectors for malware. Malware mitigation for mobile technology is a long-standing problem for which there is not yet a good solution. In this paper, we focus on identifying malicious applications, and verifying the absence of malicious or vulnerable code in applications that the enterprises and their users seek to utilize. Our analysis toolbox includes static analysis and permissions risk scoring, pre-installation vetting techniques designed to insure that malware is never installed in devices on an enterprise network. However, dynamic code-loading techniques and changing security requirements mean that apps which previously passed the verification process, and have been installed on devices, may no longer meet security standards, and may be malicious. To identify these apps, and prevent future installation of them, we propose a crowd-sourced behavioral analysis technique, using machine learning to identify malicious activity through anomalies in system calls, network behavior, and power consumption. These techniques apply effectively to single user devices over time, and to individual devices within an enterprise network.

15. SUBJECT TERMS

MAVeRiC approach to dynamic analysis for mobile-android; application security; MAVeRiC;

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Rogert A. Hallman
U	U	U	U	32	19B. TELEPHONE NUMBER *(Include area code)*: 1 619-553-7905

Standard Form 298 (Rev. 10/17)
Prescribed by ANSI Std. Z39.18

This page is intentionally blank.

This page is intentionally blank.

Distribution Statement A: Approved for public release; distribution is unlimited.

NIWC Pacific
San Diego, CA 92152-5001