NIST White Paper

Mobile Application Security Exercise (MASE)

Final Report

Michael Ogata
Software and Systems Division
Information Technology Laboratory

May 22, 2018

National Institute of
Standards and Technology
U.S. Department of Commerce

Abstract

Mobile applications have become an integral part in the mission of the federal government and public safety. There exist many techniques that seek to assure these applications are free from software bugs and vulnerabilities. However, a unified list of capabilities that define these techniques is not defined for mobile applications. This paper, through a partnership with industry, seeks to be the first set in defining this list by examining the current state-of-the-art of the analysis capabilities of mobile application vetting services.

Keywords

Apps; Applications; Mobile Apps; Mobile Application; Public Safety; Software Assurance; Vetting;

Acknowledgements

The author would like to thank the individuals and organizations that participated in the effort (in alphabetical order):

- AppCritique https://appcritique.boozallen.com
- Appthority, Inc https://www.appthority.com/
- NowSecure http://www.nowsecure.com/GO/NIST/MASE

The author would also like to thank Lolita Boumelit and Vadim Okun for their support in the formulation and execution of the MASE project. Finally, the author would like to thank Jeff Cichonski, Barbara Guttman, and Rose Linares for their help in editing and arranging the document.

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

.

Table of Contents

List of Appendices

1 Introduction

1.1 Background

The importance of mobile applications in both the public and private sector has been growing without question. Most domestic commercial mobile application stores do their best to weed out malicious applications before they are released to the public. These stores also continue to improve the labeling provided along with each mobile application that describes pertinent behaviors the mobile application expresses. However, the resilience and security requirements of both the private and public sectors are more constricting than what is provided by default in the domestic commercial mobile application stores. Furthermore, mobile applications that are purpose built for use within a sector or organization may not be placed in the public application stores, thereby avoiding any safeguards these stores provide.

These facts necessitate a review process to evaluate mobile applications for malware, software vulnerabilities, configuration vulnerabilities, and undesirable functionality. Mobile application vetting defines the process used by an organization to test that an application meets an organization's security requirements [6]. A mobile application vetting service is an entity, external to the end user's organization, that provides this mobile application testing.

Industry has risen to meet the needs of mobile application vetting. The ecosystem of strategies, methodologies, and approaches is very diverse [7]. However, comparing the strengths and weaknesses of players in this space is difficult as there does not exist a cohesive set of features that can be used to describe the capabilities and techniques of mobile application vetting services.

1.2 Purpose

The goal of the Mobile Application Security Exercise (MASE) Project is to gain a better understanding of the state-of-the-art in mobile application vetting tools. The project's main goal is to:

- Identify a list of mobile application vetting service features (capabilities) that can be used to describe the analysis capabilities of vetting services.
- Perform a preliminary and informal analysis of the mobile application analysis conducted by participating tools to gain a better understanding of the uniformity and/or cohesiveness of mobile application vetting service features among the participants.

1.3 Intended Audience

This document is intended for any organization evaluating their stance on mobile application vetting as part of their cyber security posture. The language in this document is meant for non-technical readers. As such, it may be particularly useful for organizations who are relatively new to mobile applications and mobile application vetting; especially those who may be tasked to incorporate new mobile application solutions such as the federal government and public safety.

Because of their domain knowledge, mobile application vetting service providers may also find this document useful. The field of mobile application security is nascent and evolving. NIST welcomes and encourages any input that may advance the correctness, clarity, and completeness of any of the information found in this document.

1.4 Document Structure

The remainder of this document details the MASE project. It is structured as follows:

- Project Overview – This section describes the overall process undertaken during the MASE project. It details the mobile applications used as test cases during the evaluation process of the project, the evaluation undertaken by the project's participants, and finally the process used to determine the final vetting service feature set generated by this effort.
- Mobile Application Vetting Service Features – This section defines the two features subcategories identified as part of the project. It then goes on to list and define each of the features identified.
- Mobile Application Vetting Feature Analysis – This section contains the output of the two analyses conducted as part of this exercise.
- Conclusion– This section contains final conclusions of the paper. It also identifies future work that needs to be undertaken in the space of mobile application vetting.

2 Project Overview

2.1 MASE Methodology Overview

The primary goal of the MASE is to gain a baseline understanding of the capabilities of mobile application vetting services. To achieve this goal, the exercise was devised to formulate a generalized list of analytic capabilities, from here on referred to as *features*, that can be used to describe the output of each of the vendor's solutions.

The MASE was divided into 3 phases. Phase 1 consisted of an internal NIST review of mobile application binaries to be used as test cases for the analysis phase. In Phase 2, these binaries were provided to each of the participating vetting services. The vetting services were asked to provide a single report for each application reviewed. In Phase 3 all reports were aggregated and reviewed for feature identification. Figure 1 illustrates the 3 phases of the exercise.

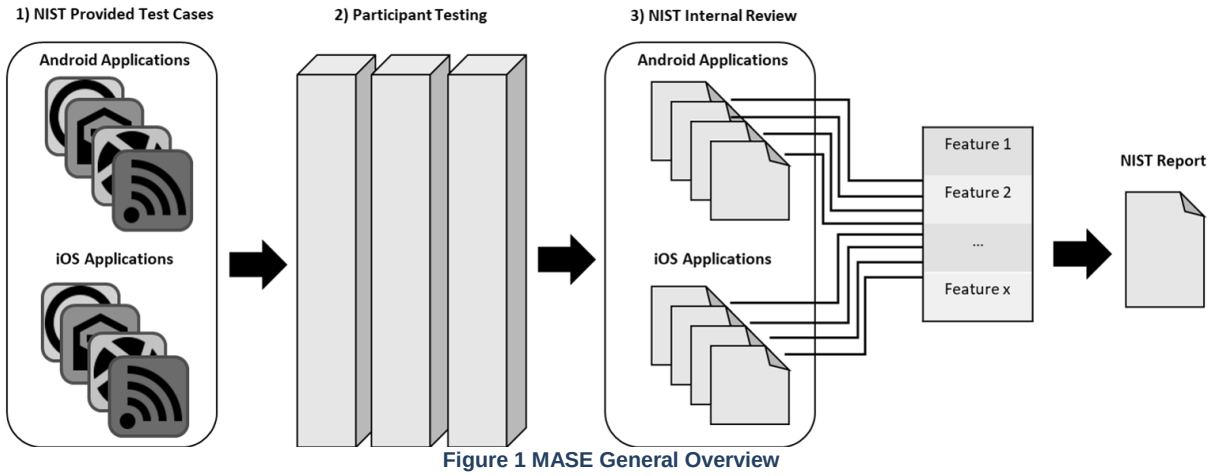

Figure 1 MASE General Overview

2.2 Test Case Overview

A set of 18 applications, 9 targeting Google Android and 9 targeting Apple iOS, was selected as the corpus for the exercise. A full list of the test cases can be found in Table 1 and Table 2. The tables assign a numeric ID to each of the test cases. For brevity, the test cases are referred to by ID for the remainder of the document.

It should be noted that there is no established ground truth for the test cases. Vetting services were analyzed for consistency. The apps in the test corpus were not independently assessed to see if a given feature is present in the app.

Table 1 Android Application Test Cases

Application Name and Version	Application ID
Facebook 46.0.0.26.153	1
GO Launcher 2.20	2
Messenger 90.0.0.14.70	3
Psafe Total 3.2.4	4
TextNow 4.27.0	5
Skype 7.18.0.505	6
Slacker Radio 6.1.27	7
Uber 3.117.3	8
Z Camera 2.37	9

Table 2 iOS Test Cases

Application Name and Version	Application ID
Google Chrome 57.0.2987.137	10
Google Drive 4.2017.10207	11
Home Station Instrumentation Training System 3.0	12
Layout from Instagram 1.2.4	13
Marriott International 6.1.1	14
OfferUp 2.10.10	15
Skype 6.34.1	16
YouTube Music 1.7	17
Twitter 6.75	18

2.3 Vendor Participation

The MASE project invited 9 mobile application vetting services to participate in the exercise. 4 vendors contributed tools for use in the project. The purpose of the exercise was not to evaluate the performance of the participants. As such, all participant names have been removed in this report except when explicitly requested by the participant.

2.4 Mobile Application Vetting Service Features

As part of the MASE project, each participant submitted a report for each application in the test case set. NIST researchers examined the submitted reports to identify and define the set of capabilities that can be

used to describe mobile application vetting analysis. Each capability, from this point referred to as a mobile application vetting service feature, acts as a label for a common reporting subject, metric, or factoid found among the mobile application reports. For this report, a feature is defined as follows:

> *A mobile application vetting service capability represented as either an analysis method/methodology or the ability to identify a particular mobile application trait*

The vetting service feature list described in this document is broken into two subsections:

- **Methods/methodology** – describes qualitative actions taken by the vetting services. For example, it was common for services to provide an itemized list of each of the permissions requested by the application.
- **Application trait** – describes identified characteristics an application exhibits. These make up the bulk of the identified service features and encapsulate things like identifying system services the application interacts with, determining if the application contains a hard-coded password, etc.

Vetting service feature compilation was a cumulative and recursive exercise. Reports were organized into sub groups, first by vendor and then by target operating system. For each subgroup, a list of features was compiled. Combining both feature lists (Android and iOS) for each vendor yielded a comprehensive list of all the features for each vendor. Finally, by combining and then normalizing each vendor's feature list, a comprehensive feature list for the exercise was generated.

2.5 Mobile Application Analysis

Once the feature list was extracted, two comparisons were conducted:

Analysis 1: Feature Vendor Coverage – This analysis illustrates the frequency with which the identified mobile application vetting service features occurred among the participating tool vendors.

Analysis 2: Application Trait Identification Per Application – This analysis describes the number of participating vendors that agree on the existence of a mobile application vetting service feature within a mobile application test case. It is meant to illustrate consensus and disagreements about feature existence between vendors on a per app level.

3 Mobile Application Vetting Service Features

This section provides a detailed list and definitions of the two subtypes of mobile application vetting service feature identified in the MASE project: Analysis Methods/Methodologies and Application Traits.

3.1 Analysis Methods and Methodologies

Feature Name	Feature Definition
App Integrity Measure	The application's certificate and/or digital signature are valid.
Assigns App Score	The application vetting service applies either a quantitative or qualitative score to the application. This application score is meant to represent the degree to which an application is free from vulnerabilities and/or threats.

Feature Name	Feature Definition
Enumerates App Permission Requests	The application vetting service enumerates application permissions requested by the application as part of the application report.
Enumerates Package Filenames	The mobile application vetting service enumerates all files associated with the application as part of the application report.
Evaluates TLS operations	The mobile application vetting service evaluates the application's use of transport layer security. This can include: • Evaluating the application's use of certificate pinning • Evaluating the application's acceptance criteria for hostnames, certificates, and/or certificate authorities
Extracts raw string data	The mobile application vetting service enumerates human readable text as part of the application report.
Flags potential PII exposure	The mobile application vetting service explicitly identifies the application's potential to expose Personally Identifiable Information (PII).
Flags Potentially Undesirable Behavior	As part of the application report, the mobile application vetting service identifies behavior that, while not an explicit weakness or vulnerability, may be considered against best practice or risky.
Provides App Descriptive Metadata	The application vetting service provides app metadata either extracted from the application binary itself or from the application's origin app store as part of the report.
References Official Regulatory Classifier: CVSS	The mobile application vetting service maps vulnerabilities, completed diagnostics, and/or other reporting facets to the Common Vulnerability Scoring System (CVSS) (Common Vulnerability Scoring System SIG, n.d.).
References Official Regulatory Classifier: CWE	The mobile application vetting service maps vulnerabilities, completed diagnostics, and/or other reporting facets to the Common Weakness Enumeration (CWE) dictionary (Common Weakness Enumeration, n.d.).
References Official Regulatory Classifier: NIAP	The mobile application vetting service maps vulnerabilities, completed diagnostics, and/or other reporting facets to National Information Assurance Partnership Protection (NIAP) Profiles (U.S. Government Approved Protection Profile - Protection Profile for Application Software Version 1.2, 2016).
References Official Regulatory Classifier: OWASP	The mobile application vetting service maps vulnerabilities, completed diagnostics, and/or other reporting facets to Open Web Application Security Project (OWASP) Mobile Top Ten Mobile Risks (OWASP Mobile Security Project - Top 10 Mobile Risks, 2017).

Feature Name	**Feature Definition**
Reports on network traffic	The mobile application vetting service reports on observed network traffic as part of the mobile application report.

3.2 Application Traits

Feature Name	*Feature Definition*
Accesses battery Information	The application requests permission to interrogate the operating system concerning the current state of the battery.
Accesses Device Identifier	The application accesses a unique identifier of the device including • Device's name • Universally Unique Identifier (UUID) • Unique Device Identifier (UDID) • International Mobile Subscriber Identity (IMSI) • Subscriber Identity Module (SIM) serial number
Accesses fingerprint information	The application requests permission to access the device's fingerprint subsystem.
Application Backup Interaction	The application interacts with backup systems provided by the mobile operating system. This behavior indicates the application has the potential to export data to external sources.
Cryptographic Issues (Network)	The application fails to ensure data is transmitted in properly encrypted channel (see *Cryptography Issues* for potential causes).
Cryptographic Issues (Storage)	The application fails to ensure data stored on the device is properly encrypted (see *Cryptography Issues* for potential causes).
Cryptography Issues	The application contains weaknesses or flaws that may affect cryptography operations on the device. This can be the result of, but are not limited to: • Lack of encryption • The detection of weak or broken ciphers and algorithms • The detection of misconfigured random number generators
Detects Debugging Status	The application is configured to be examinable by a debugger. Distributing an application in such a manner may make it easier to reverse engineer and is against best practice.
Detects filesystem problems	The application exhibits the potentially risky behavior of creating or modifying files that can be read or written to by other processes on the device. This behavior has the potential to jeopardize both the confidentiality and integrity of data residing within the application. Furthermore, it can expose the inner structure of an application and enable easier reverse engineering. This feature does not describe the ability of the application to maliciously access files of other applications or the operating system.

Feature Name	*Feature Definition*
Detects hardcoded credentials	The application stores passwords and/or cryptographic keys in an accessible form within the application.
Detects if app is over-permissioned	The application requests more functionality than is used.
Detects Insecure Password Practices	The application fails to enforce best practices when requiring the use of a password. This can be the result of: • The use of passwords that fail to meet complexity and/or length requirements • The detection of passwords transmitted by the application in plain text.
Detects Jailbreak/Root	The application has the capability to detect if the device on which it is running has been jailbroken.
Detects Native Code	The application has been packaged with binary C or C++ executables and/or libraries
Detects Unsafe Compilation Settings	The application has been compiled using parameters that may introduce weaknesses and vulnerabilities in the resultant application binary.
Dynamically loads code (Android)	The application is constructed such that it may be possible to obtain new, unvetted functionality from unknown sources while the application is running.
Dynamically loads code (JavaScript)	The application is constructed such that it can obtain new, unvetted functionality in the form of remotely obtained JavaScript.
Examines/interacts with other applications	The application requests some or all the following functionality: • Starting or stopping other applications • Examining what applications are currently installed on the device • Examining what applications are currently running on the device • Sending data to other applications via inter-application systems defined by the operating system.
Executes Subshell (Android)	The application executes operating system commands via a shell.
Executes system level commands (ios)	The application executes low level or kernel level system commands.
Exports system runtime information	The application exports app crash data to third party analytic entities.
Identifies ad network connections	The application has the capability to make remote connections to a known ad network(s).

Feature Name	*Feature Definition*
Identifies cloud storage service connections	The application has the capability to make remote connections to a known cloud storage service(s).
Identifies Connections to Foreign Countries	The application vetting service enumerates network connections made to foreign countries.
Identifies explicit or named vulnerabilities	The application vetting service makes explicit positive or negative assertions as to the existence of known, named, and well-characterized security vulnerabilities. This could include named groups of vulnerabilities such as Stagefright or classes of weaknesses such as structured query language (SQL) Injection.
Identifies Keychain Issues (iOS)	The application is detected to use the device Keychain in an unsafe manner.
Identifies social network interaction	The application has the capability to make remote connections to a known social network(s).
Identifies specific known malware	The application vetting service makes explicit positive or negative assertions as to the existence of known, named, mobile malware.
Identifies third party analytic connections	The application has the capability to make remote connections to a third-party usage analytic service.
Identifies third party libraries and APIs	The application vetting service enumerates all detected third-party programming libraries included in the application.
Identifies VPN Functionality	The application has the capability to interact with virtual private network (VPN) functionality on the device.
Implements Memory Protection (iOS)	The application has been compiled using memory protection safeguards.
Interacts with Apple Watch (iOS)	The application interacts with the Apple Watch peripheral.
Interacts with Bluetooth	The application requests access to the Bluetooth radio on the device.
Interacts with device accounts	The application requests access to read or modify device user account information
Interacts with device calendar	The application requests access to read/write to/from the device calendar.
Interacts with device camera	The application requests access to the device camera.
Interacts with device contact list	The application requests access to read/write to/from the contact list.
Interacts with device health API	The application requests access to health data provided by the device.
Interacts with device microphone	The application requests access to the device microphone.

Feature Name	*Feature Definition*
Interacts with device photo storage	The application requests access to the device photo library.
Interacts with device telephony service	The application requests some or all the following capabilities: • Making/accepting phone calls • Examining the status of an ongoing call
Interacts with external storage	The application can read and/or write to external sources on the device. This may include removable storage locations.
Interacts with location services	The application is noted to explicitly access one or more of the location services provided by the device.
Interacts with Near Field Communication (NFC) Radio	The application requests access to the device NFC radio.
Interacts with Short Message Service (SMS)/Multimedia Messaging Service (MMS) Services	The mobile application requests some or all the following capabilities: • Send SMS/MMS messages • Read the contents of SMS/MMS messages • Write/modify the contents of an SMS/MMS message
Interacts with system logs	The mobile application can read or write from the system logs on the device.
Interacts with USB interface	The application has the capability to connect devices or peripherals via the USB interface.
Interacts with Wi-Fi connections	The application has requested the ability to monitor or change the state of Wi-Fi radio.
References Official Regulatory Classifier: CVE	The mobile application vetting service maps vulnerabilities, completed diagnostics, and/or other reporting facets to the Common Vulnerability Enumeration (CVE) database (National Vulnerability Database, n.d.).
Requests device admin functionality	The application explicitly requests the ability to assume administrative control over the device.
Requests Internet Access	The application explicitly requests the ability to access the internet.

4 Mobile Application Vetting Feature Analysis

The primary goal of the MASE is to gain a better understanding of the capabilities of mobile application vetting services. To further this goal, using the features identified in sections 3.1 and 3.2, two analyses were conducted. The first analysis counts the number of tools capable of identifying various mobile app features. This analysis is detailed in section 4.1 The second counts the number of tools which identified the features in our test cases. Note: the results show tool overlap, not ground truth. The results of this analysis are

detailed in 4.2

4.1 Analysis 1: Feature Vendor Coverage

Understanding the frequency with which a feature is represented among the participating tools helps to illustrate how common a tool capability is when looking at the state of the art. Table 3 contains a count of the number of tools capable of identifying the methods/methodologies identified in 3.1. Likewise, Table 4 contains a count of the number of tools capable of identifying the application traits identified in section 3.2 Each table subdivides its results by operating system. For each table, the maximum value that can be found for each corpus corresponds to the number of vendors that can analyze applications in that category: 4 for Android and 3 for iOS.

In terms of identified methods and methodologies, there was a reasonable amount of homogeneity among the analysis tools, especially in the Android tools. Enumerating app permission requests is the most prevalent analysis methodology, however application integrity methods, extracting raw string data, reporting on network traffic, and evaluating TLS operation were also commonly employed. Vendors evaluating the iOS corpus focused fewer times on enumerating app permissions, opting more to focus on network analysis and TLS operations. The most disparity at the vendor level occurred with the frequency in which they referred to any of the noted regulatory classifiers. In this vein, CVSS, NIAP, and OWASP were favored over CVE and CWE.

Examining how many vendors counted each of the applications' traits reveals which traits were common capabilities among the participants. Figure 2 summarizes the distribution of how many application features were discoverable by how many (0 to 4) of the participants. For instance, there were 12 iOS features that no tools claimed to be able to identify and 3 that were identified by 3 tools. There were no features that could be identified by all tools.

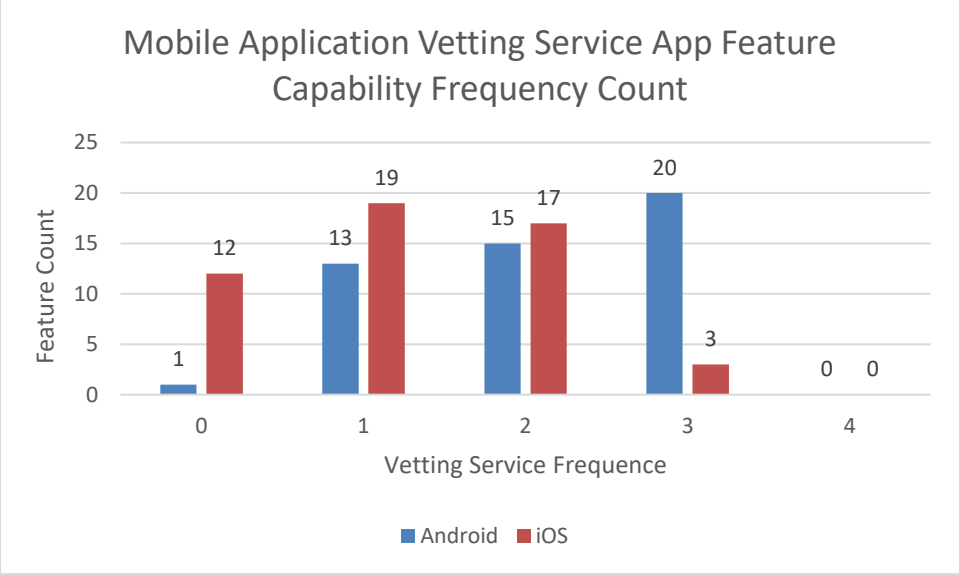

Figure 2 Mobile Application Vetting Service App Feature Capability Frequency Count

Table 3 Vendor Count of Represented Features: Methods and Methodologies

Methods and Methodologies	# of Vendors expressing Feature Capability	
	Android Corpus	iOS Corpus
App Integrity Measure	3	1
Assigns App Score	2	2
Enumerates App Permission Requests	4	1
Enumerates Package Filenames	2	2
Evaluates TLS operations	3	3
Extracts raw string data	3	2
Flags Potentially Undesirable Behavior	4	3
Provides App Descriptive Metadata	3	2
References Official Regulatory Classifier: CVE	1	0
References Official Regulatory Classifier: CVSS	2	2
References Official Regulatory Classifier: CWE	1	1
References Official Regulatory Classifier: NIAP	2	1
References Official Regulatory Classifier: OWASP	2	2
Reports on network traffic	3	2

Table 4 Vendor Count of Represented Features: Application Traits

Application Trait	# of Participants Expressing Feature Capably	
	Android	iOS
Accesses battery Information	2	0
Accesses Device Identifier	3	1
Accesses fingerprint information	1	1
Application Backup Interaction	1	1
Cryptographic Issues (Network)	3	3
Cryptographic Issues (Storage)	2	1
Cryptography Issues	3	3
Detects Debugging Status	2	1
Detects filesystem problems	2	1
Detects hard coded credentials	3	2
Detects if app is over permissioned	1	0
Detects Insecure Password Practices	2	2
Detects Jailbreak/Root	1	2

Application Trait	# of Participants Expressing Feature Capably	
	Android	iOS
Detects Native Code	2	1
Detects Unsafe Compilation Settings	0	2
Dynamically loads code (Android)	3	
Dynamically loads code (JavaScript)	2	0
Examines/interacts with other applications	3	1
Executes as root	1	0
Executes Subshell (Android)	2	
Executes system level commands (iOS)		2
Exports system runtime information	1	1
Identifies ad network connections	3	1
Identifies cloud storage service connections	2	2
Identifies Connections to Foreign Countries	2	2
Identifies explicit or named vulnerabilities	3	2
Identifies Keychain Issues (iOS)		1
Identifies potential PII exposure	1	1
Identifies social network interaction	3	2
Identifies specific known malware	1	2
Identifies third party analytic connections	2	1
Identifies third party libraries and APIs	3	2
Identifies VPN Functionality	1	1
Implements Memory Protection (iOS)		1
Interacts with Apple Watch (iOS)		1
Interacts with Bluetooth	3	2
Interacts with device accounts	2	0
Interacts with device calendar	3	2
Interacts with device camera	3	2
Interacts with device contact list	3	3
Interacts with device health API	1	1
Interacts with device microphone	3	2
Interacts with device photo storage	1	0
Interacts with device telephony service	3	1
Interacts with external storage	2	0
Interacts with location services	3	2
Interacts with Near Field Communication (NFC) Radio	2	0
Interacts with SMS/MMS Services	3	2
Interacts with system logs	3	0
Interacts with USB interface	1	0

Application Trait	# of Participants Expressing Feature Capably	
	Android	iOS
Interacts with Wi-Fi connections	2	0
Requests device admin functionality	1	0
Requests Internet Access	3	1

4.2 Analysis 2: Application Trait Identification Per Application

Analysis 2 describes how many vendors identified the existence of each trait within each of the application test cases. All 4 participating vendors supported analysis for Android applications. Therefore, the largest possible count for each of the application traits in the Android corpus (App IDs 1 to 9) is 4. There is an exception however for App ID 3, as the report for this application was damaged and could not be included in the analysis for one participating vendor. This renders the maximum value for this column 3. All but one of the participating vendors had the capability to analyze iOS applications. Therefore, the largest possible count for each of the application traits in the iOS corpus (App IDs 10 to 18) is 3. Table 6 captures the results of this analysis.

There are two conclusions that can be made from this analysis. The first shows the amount of consensus between the participants concerning the existence of an application trait[1]. There are two values that represent consensus, the first being the consensus that an application trait is *not* expressed by an application which is represented by a count of 0. The second value indicates a consensus that an application trait *is* represented by an application. This value is represented by a count of 4 for Android and 3 for iOS[2]. Any value other than these represents disparity amongst the mobile application vetting services concerning the existence of an application trait.

The second conclusion shown by the analysis conveys the frequency of application traits as they appear in the data set.

Among the test case set the following attributes were observed to occur with the highest frequency within the application corpuses:

Table 5 Ten Highest Observed Frequency Application Traits

Highest Frequency Android Application Traits	Highest Frequency iOS Application Traits
Requests Internet Access	Identified explicit or named vulnerabilities
Interacts with location services	Detects Debugging Status
Cryptography Issues	Interacts with location services
Interacts with device telephony service	Interacts with device camera
Interacts with device contact list	Interacts with device contact list
Interacts with SMS/MMS Services	Cryptographic Issues (Network)
Interacts with device camera	Requests Internet Access
Identifies social network interaction	Detects Native Code
Interacts with device microphone	Identifies ad network connections
Interacts with device accounts	Detects hard coded credentials

[1] An important note concerning the analysis: the count associated with each application trait and application is not the count of the number of occurrences of said trait within the application. Rather, it is the count of the number of vendors that identified a trait's existence within a given application.

[2] Note, a full consensus does not exist in the data set for either corpuses

Table 6 Vendor Application Traits Counts Grouped by Application

Application Traits	Android Corpus									Android Sums	iOS Corpus									iOS Sums
	1	2	3	4	5	6	7	8	9		10	11	12	13	14	15	16	17	18	
Accesses battery Information	2	0	2	2	2	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
Accesses Device Identifier	2	2	2	2	2	1	1	3	2	17	1	1	0	1	1	1	1	1	1	8
Accesses fingerprint information	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	3
Application Backup Interaction	1	1	1	1	1	1	1	1	0	8	1	0	0	0	0	1	0	0	1	3
Cryptographic Issues (Network)	1	2	1	2	3	1	1	2	2	15	1	1	2	0	0	2	1	2	2	11
Cryptographic Issues (Storage)	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Cryptography Issues	3	3	3	3	3	3	3	3	2	26	0	0	0	0	0	0	0	0	0	0
Detects Debugging Status	1	1	1	1	1	1	1	1	0	8	2	2	0	0	1	2	2	2	2	13
Detects filesystem problems	1	1	1	1	1	1	1	1	0	8	0	0	0	0	0	0	0	0	0	0
Detects hard coded credentials	0	1	0	2	1	0	2	1	1	8	1	1	1	1	1	1	1	1	1	9
Detects if app is over permissioned	1	1	1	1	1	1	1	1	0	8	0	0	0	0	0	1	0	0	0	1
Detects Insecure Password Practices	0	0	0	1	2	1	2	2	0	8	0	0	0	0	0	0	0	0	0	0
Detects Jailbreak/Root	0	0	0	0	1	0	0	0	1	2	0	0	0	0	0	1	0	1	1	3
Detects Native Code	2	2	2	2	2	2	2	2	2	18	1	1	1	1	1	1	1	1	1	9
Detects Unsafe Compilation Settings	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Dynamically loads code (Android)	2	2	2	3	2	1	2	2	1	17										
Dynamically loads code (JavaScript)	1	2	2	1	2	2	2	2	1	15	0	0	0	0	0	0	0	0	0	0
Examines/interacts with other applications	2	3	1	2	2	2	2	1	3	18	1	0	1	1	0	1	0	1	1	6
Executes as root	0	0	0	0	1	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0
Executes Subshell (Android)	1	2	1	2	1	1	1	1	2	12										
Executes system level commands (iOS)										0	1	1	0	1	1	1	1	1	0	7
Exports system runtime information	0	1	0	0	1	0	0	1	0	3	0	0	0	0	0	1	0	0	1	2

Application Traits	Android Corpus									Android Sums	iOS Corpus									iOS Sums
	1	2	3	4	5	6	7	8	9		10	11	12	13	14	15	16	17	18	
Identified explicit or named vulnerabilities	1	1	1	1	1	1	1	1	0	8	2	3	1	1	3	3	3	3	2	21
Identifies ad network connections	0	2	0	2	3	2	3	2	2	16	1	1	1	1	1	1	1	1	1	9
Identifies cloud storage service connections	0	1	0	0	1	0	0	0	1	3	1	0	0	0	0	1	0	1	0	3
Identifies Connections to Foreign Countries	1	1	1	1	1	1	1	1	1	9	0	1	0	0	0	0	1	0	0	2
Identifies Keychain Issues (iOS)										0	1	0	1	1	1	1	1	1	1	8
Identifies potential PII exposure	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	5
Identifies social network interaction	2	2	1	3	3	1	2	3	3	20	1	1	0	1	0	1	1	1	1	7
Identifies specific known malware	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Identifies third party analytic connections	0	2	0	1	2	1	1	0	1	8	0	0	0	1	2	1	1	1	0	6
Identifies third party libraries and APIs	1	1	1	1	2	1	1	2	1	11	0	0	0	0	0	0	0	0	0	0
Identifies VPN Functionality	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
Implements Memory Protection (iOS)										0	1	1	1	1	1	1	1	1	1	9
Interacts with Apple Watch (iOS)										0	0	0	0	0	1	0	1	0	0	2
Interacts with Bluetooth	0	3	3	3	3	3	3	0	0	18	2	0	0	0	2	0	0	0	0	4
Interacts with device accounts	2	2	2	2	2	2	2	2	2	18	0	0	0	0	0	0	0	0	0	0
Interacts with device calendar	3	0	0	3	0	0	0	0	0	6	0	2	0	0	2	0	2	0	0	6
Interacts with device camera	3	3	3	3	0	3	1	3	3	22	2	1	0	1	1	2	1	2	2	12
Interacts with device contact list	3	3	3	3	3	3	3	3	1	25	0	2	0	2	0	1	3	2	1	11
Interacts with device health API	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Interacts with device microphone	3	1	3	1	3	3	3	0	3	20	2	0	0	0	0	1	2	2	1	8
Interacts with device photo storage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Interacts with device telephony service	2	3	3	3	3	3	3	3	3	26	0	0	0	0	0	0	0	0	0	0
Interacts with external storage	2	2	2	2	2	2	2	2	2	18	0	0	0	0	0	0	0	0	0	0
Interacts with location services	3	3	3	3	3	3	3	3	3	27	2	0	2	1	2	2	2	0	1	12

Application Traits	Android Corpus									Android Sums	iOS Corpus									iOS Sums
	1	2	3	4	5	6	7	8	9		10	11	12	13	14	15	16	17	18	
Interacts with Near Field Communication (NFC) Radio	0	0	2	0	0	0	2	0	0	4	0	0	0	0	0	0	0	0	0	0
Interacts with SMS/MMS Services	3	3	3	3	3	3	1	3	1	23	0	0	0	1	0	2	1	1	0	5
Interacts with system logs	0	3	3	2	3	0	1	0	0	12	0	0	0	0	0	0	0	0	0	0
Interacts with USB interface	0	0	0	0	0	0	1	1	0	2	0	0	0	0	0	0	0	0	0	0
Interacts with Wi-Fi connections	2	2	2	2	2	2	2	2	2	18	0	0	0	0	0	0	0	0	0	0
Requests device admin functionality	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Requests Internet Access	3	3	3	3	3	3	3	3	3	27	1	1	1	1	1	1	1	1	1	9

5 Conclusion

Mobile application vetting services have a wide range of capabilities. The MASE project identified 67 features of mobile app vetting services that describe their capabilities. The project divided these features into two subsets: Methods/Methodologies mobile application vetting services employ to analyze and describe their findings; and application traits that describe the existence of a characteristic intrinsic to the application.

The features used to analyze mobile applications vary from vendor to vendor. Furthermore, the features used to analyze Android and iOS application differ slightly even for the same vendor. However, generally the vendors focus their methodologies on:

- Enumerating application permissions
- Evaluating network traffic
- Evaluating TLS operations
- Extracting raw string data from the application binary

Many of the commonly identifiable application attributes in the participants' repertoire included identifying when an application interacted with an operating system service (calendar, contact list, photo storage, SMS, etc.) or device peripheral (cameras, Bluetooth radios, external storage, etc.). Detecting the existence of hard coded credentials was also a common attribute of mobile application vetting services.

5.1 Recommendations and Future Work

5.1.1 Building mobile application test cases

Ascertaining the true capabilities of a mobile application vending service requires first having ground truth concerning what is being analyzed. Currently, a robust set of mobile application test cases for use in statistically analyzing vetting service performance does not exist. Due to the diversity in mobile platforms and speed of change of these platforms, building and maintaining a set of mobile applications for use as test cases is a challenging task. It is the recommendation of the MASE project that the organization(s) set out to build and maintain such a set to further the evaluative capacity of mobile application vetting services in the future.

5.1.2 Software Assurance Tool Exposition (SATE) for Mobile

The Software Assurance Tool Exposition (SATE) is a noncompetitive exercise maintained and run by the Software Assurance Metrics and Tool Evaluation (SAMATE) group at the National Institute of Standard and Technology (NIST). It is designed to advance the state-of-the-art in tools that find security-related defects in software applications. As of the time of this writing, preparations are underway for the sixth iteration of the SATE. This marks the first iteration with a sub-track, specifically targeting mobile applications. This track represents a logical extension to the efforts made as part of the MASE as it will focus on the rate with which mobile application vetting services successfully identify vulnerabilities intentionally seeded into a set of mobile application [8].

Appendix A - Glossary

Application Permission	In the current application developer paradigm, applications must request access to data and services provided by either the application's host operating system or other applications that live along-side them. An application permission represents a declarative request made by the developer of an application to be granted access to said data or service [9] [10].
Feature	A mobile application vetting service capability represented as either an analysis method/methodology or the ability to identify a particular mobile application trait
Mobile Application Vetting	The process of verifying that an app meets an organization's security requirements. An app vetting process comprises app testing and app approval/rejection activities (Quirolgico, Voas, Karygiannis, Michael, & Scarfone, 2015).
Mobile Application Vetting Service	An entity that engages in the mobile application vetting process on behalf of another organization.

Appendix B - References

[1] "National Vulnerability Database," National Institute of Standards and Technology (NIST), [Online]. Available: https://nvd.nist.gov/.

[2] "Common Vulnerability Scoring System SIG," Forum of Incident Response and Security Teams (FIRST), [Online]. Available: https://www.first.org/cvss/.

[3] "Common Weakness Enumeration," MITRE Corporation, [Online]. Available: https://cwe.mitre.org/.

[4] "U.S. Government Approved Protection Profile - Protection Profile for Application Software Version 1.2," National Information Assurance Partnership (NIAP), 22 04 2016. [Online]. Available: https://www.niap-ccevs.org/pp/pp_app_v1.2.htm.

[5] "OWASP Mobile Security Project - Top 10 Mobile Risks," Open Web Applicatio Security Project (OWASP), 27 April 2017. [Online]. Available: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks.

[6] S. Quirolgico, J. Voas, et. al, *Vetting the Security of Mobile Applications*, NIST Special Publication (SP) 800-163, National Institute of Standards and Technology, Gaithersburg, Maryland, June 2015, 37pp. http://dx.doi.org/10.6028/NIST.SP.800-163.

[7] G Howel and M. Ogata, *Mobile Application Vetting Services for Public Safety*, NISTIR 8136, National Institute of Standards and Technology, Gaithersburg, Maryland, June 2016, 13pp. https://doi.org/10.6028/NIST.IR.8136.

[8] Static Analysis Tool Exposition VI (SATE VI), Software Assurance Metrics and Tool Evaluation (SAMATE) [Website], https://samate.nist.gov/SATE6.html [accessed 12/07/17].

[9] System Permissions, Google Developer Program [Website], https://developer.android.com/guide/topics/permissions/index.html [accessed 12/07/17].

[10] Requesting Permissions, Apple Developer Program [Website], https://developer.apple.com/ios/human-interface-guidelines/app-architecture/requesting-permission/ [accessed 12/07/17].