Mobile Working Group

Mobile Application Security Testing Initiative

June 2016

White Paper

CSA cloud security alliance®

The permanent and official location for Cloud Security Alliance Mobile research is https://cloudsecurityalliance.org/group/mobile/

TABLE OF CONTENTS

Acknowledgments 1

Executive Summary 2

1. Introduction 4

- 1.1 Purpose and Scope
- 1.2 Normative References
- 1.3 Preliminary Study
- 1.4 Structure of this Paper

2. Problem Statement 8

- 2.1 Mobile Computing and Application Security Challenges
- 2.2 Third-Party Application-Derived Security Challenges
- 2.3 Mobile Application Development Management Challenges
- 2.4 Mobile Application Security Vetting Concerns
  - 2.4.1 Intentional Misconduct
  - 2.4.2 Negligence
  - 2.4.3 Native Problems

3. Mobile Application Security Management Lifecycle 13

4. CSA Mobile Application Security Testing Scheme 17

- 4.1 Types of Security Vetting
- 4.2 Mobile Application Security Requirements
  - 4.2.1 Privacy Handling – Permission Misuse
  - 4.2.2 Privacy Handling – Improper Information Disclosure
  - 4.2.3 Native Security – API/LIB Native Risk
  - 4.2.4 Native Security – Application Collusion Activity
  - 4.2.5 Native Security – Development Obfuscation Concern
  - 4.2.6 Protection Requirement – Connection Encryption Strength
  - 4.2.7 Protection Requirement – Data Storage
  - 4.2.8 Execution Environment – Power Consumption
  - 4.2.9 Summary
- 4.3 Mobile Application Security Vetting Methodology
  - 4.3.1 Definition, Requirements and Objectives
  - 4.3.2 Content Classification and Rating
  - 4.3.3 Steps and Procedures

Bibliographic Citations 28

ACKNOWLEDGMENTS

(In alphabetical order)

Co-Chairs

Douglas Lee

Eric Wang

Contributors

Edward Chiu

Aaron Guzman

Dong Ji

Curtis Kozielec

Setumadhav Kulkarni

Keng Lee

Zhan Leilei

Yin Liping

Dean McBride

Porus Mehta

Srinivas Naik

Timothy G. O'Brien

Mark Perry

Jim Pinter

VijayVK Velu

Matt Wehnes

Special Thanks

Timothy Grance

Stephen Quirolgico

CSA Staff

Mickey Law, Research Analyst

Lynne Yang, Research Analyst

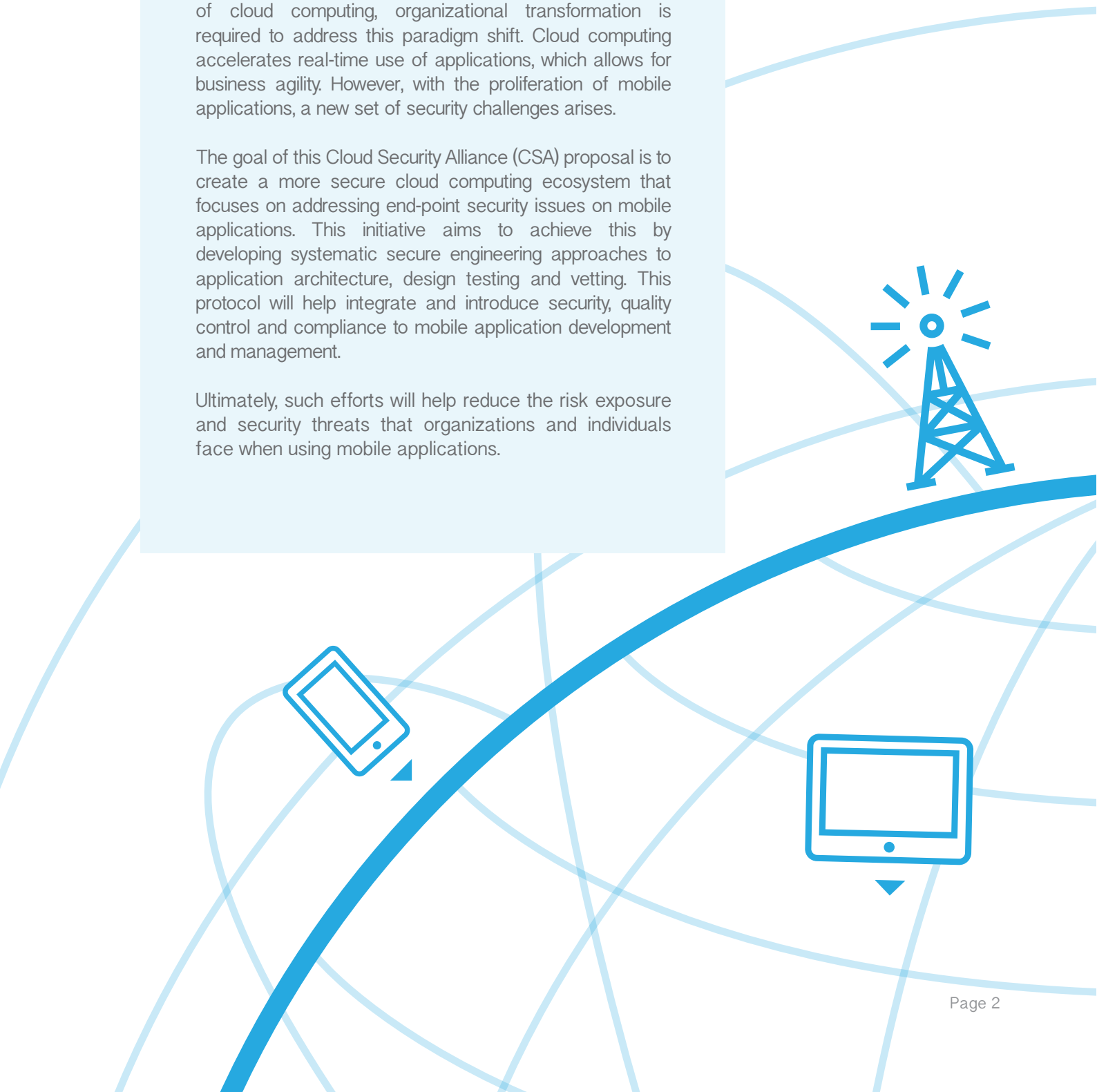
John Yeoh, Senior Research Analyst

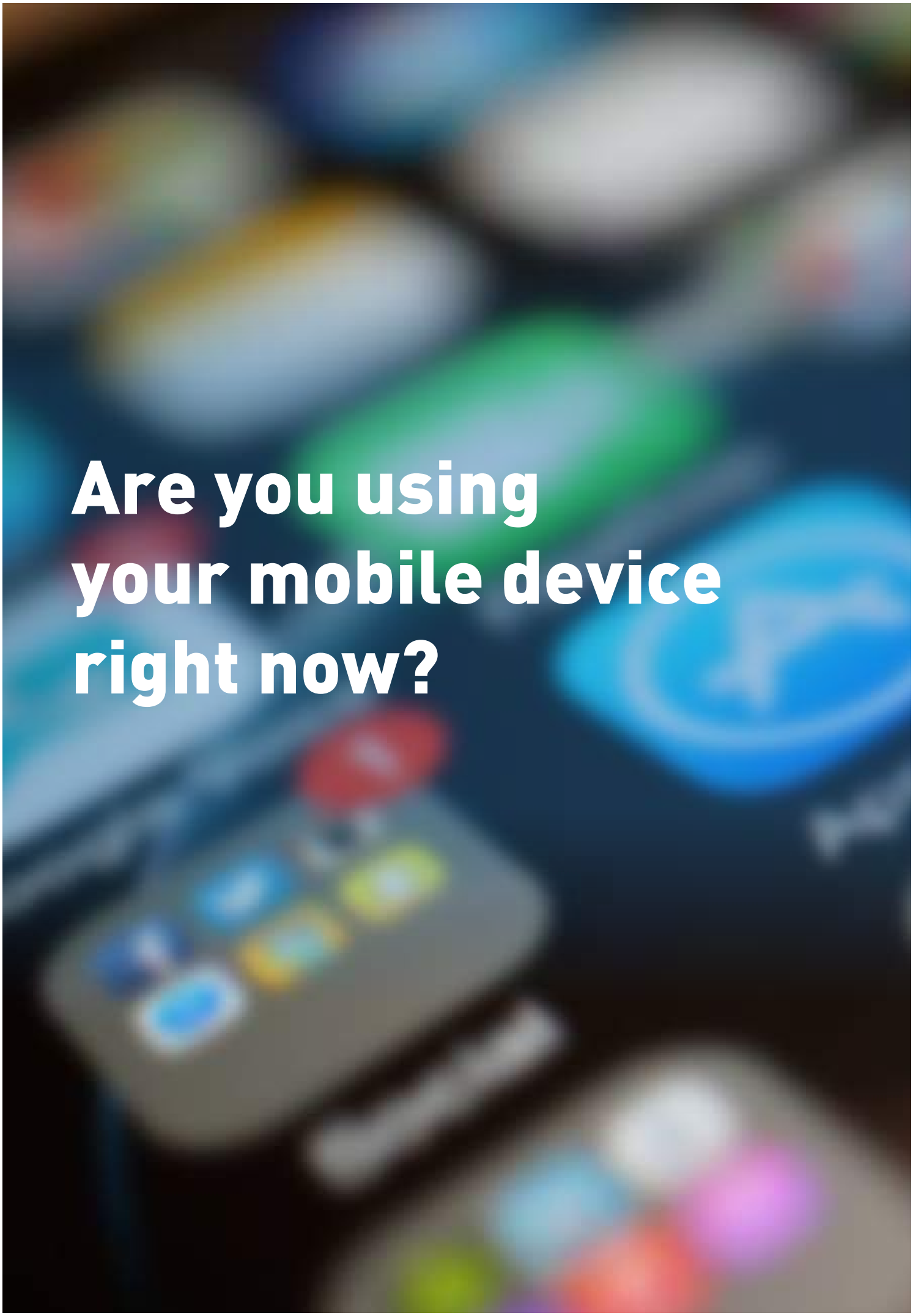
Executive Summary

Mobile applications have not only become indispensable components of our daily lives, they have also become integral parts of organizational culture. With the emergence of cloud computing, organizational transformation is required to address this paradigm shift. Cloud computing accelerates real-time use of applications, which allows for business agility. However, with the proliferation of mobile applications, a new set of security challenges arises.

The goal of this Cloud Security Alliance (CSA) proposal is to create a more secure cloud computing ecosystem that focuses on addressing end-point security issues on mobile applications. This initiative aims to achieve this by developing systematic secure engineering approaches to application architecture, design testing and vetting. This protocol will help integrate and introduce security, quality control and compliance to mobile application development and management.

Ultimately, such efforts will help reduce the risk exposure and security threats that organizations and individuals face when using mobile applications.




Are you using
your mobile device
right now?

1 Introduction

1.1 Purpose and Scope

Cloud Security Alliance (CSA) Mobile Application Security Testing (MAST) aims to define a framework for secure mobile application development, achieving privacy and security by design. Implementation of MAST will result in clearly articulated recommendations and best practices in the use of mobile applications.

Mobile application security testing and vetting processes utilized through MAST involve both static and dynamic analyses to evaluate security vulnerabilities of mobile applications for platforms such as Android, iOS and Windows. These processes cover permissions, exposed communications, potentially malicious functionalities, application collusions, obfuscations, excessive power consumptions and traditional software vulnerabilities. Testing and vetting processes will also cover internal communications such as debug flag and activities, as well as external communications such as Global Positioning System (GPS), Bluetooth, Near Field Communication (NFC) and Global System for Mobile communication (GSM) accesses [9]. Apart from mobile application security testing and vetting, a mobile application security incident response plan will also be developed.

This document references NIST Special Publication 800-163 as the basis of consideration in determining classification levels for basic security vetting specifications [9]. These benchmarks provide third-party institutions with related application security vetting, vetting result analysis, security risk

assessments for mobile applications and the respective security level ratings, thereby greatly enhancing mobile application security. Furthermore, this whitepaper outlines necessary security vetting requirements and baselines for mobile applications.

1.2 Normative References

- NIST Special Publication 800-115, Technical Guide to Information Security Testing and Assessment [5]
- NIST Special Publication 800-142, Practical Combinatorial Testing [3]
- NIST Special Publication 800-163, Vetting the Security of Mobile Applications [9]
- Cloud Security Alliance, Security Guidance for Critical Areas of Mobile Computing [1]
- Cloud Security Alliance, Top Threats to Mobile Computing [2]
- OWASP Mobile Security Project [8]
- Ministry of Industry and Information Technology of the People's Republic of China, YD/T 2407-2013: Technical requirements for security capability of smart mobile terminal [6]
- Ministry of Economic Affairs (Taiwan), Mobile Application Security Vetting Specifications [7]
- International Organization for Standardization, ISO/IEC 7034:2011: Information technology — Security techniques — Application security [4]

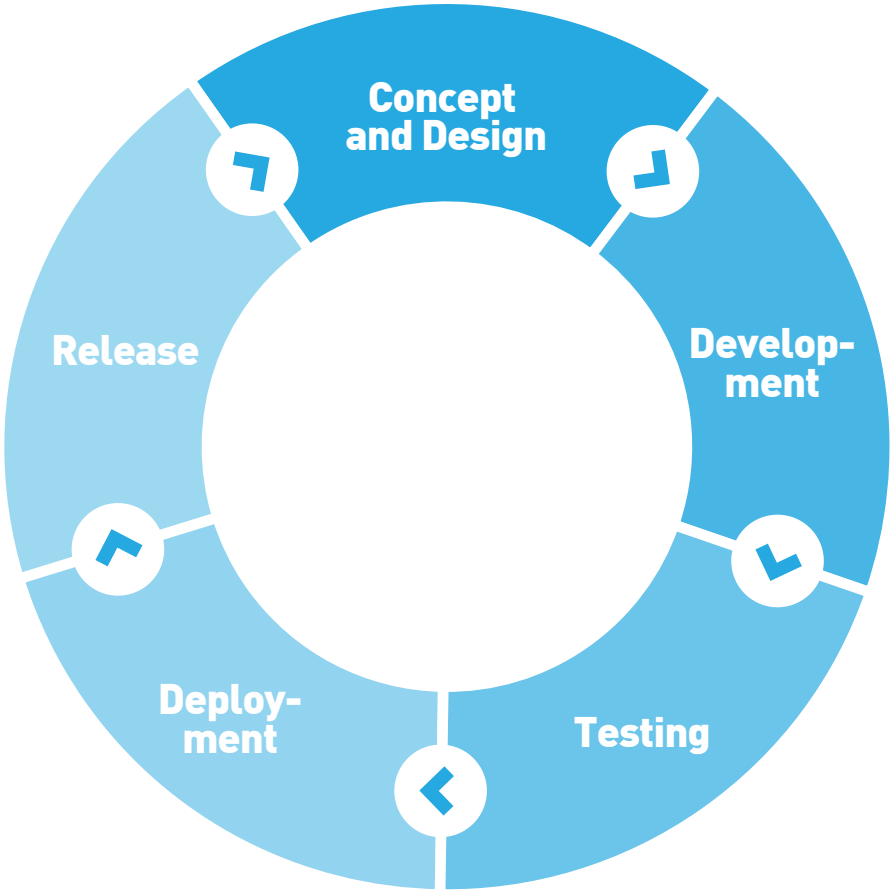
1.3 Preliminary Study

CSA has identified that privacy-related issues are going to be the key areas of concern in this initiative. Therefore, avoiding improper disclosure of information should be a basic requirement for mobile applications.

Privacy-related vulnerabilities and risks are often embedded in the application during its development, with this action either performed out of malice or negligence. Additionally, most mobile applications initiate connections to websites or services outside of the application. Therefore, mobile data encryption or related data protection controls should be taken into account during the application development phase.

Mobile devices have a much shorter uptime compared to traditional portable computing devices, due to frequent usage and limited battery power capacity. As such, energy consumption requirements and potential related problems should be considered during development. Application energy demands should be limited to only what is needed for successful application service delivery.

Mobile Application Lifecycle



Conventionally, testing should be carried out immediately after the development phase—before distribution—in a mobile application lifecycle. CSA believes that by achieving privacy and security by design, and by verifying the security of a mobile application before its distribution, the interest of application users can be best protected.

1.4 Structure of this Paper

This whitepaper covers four major sections.

Section 1 introduces purpose and scope; normative references; a preliminary study; and outlines the structure of this whitepaper.

Section 2 examines the key areas of concern and challenges during mobile application security vetting.

Section 3 discusses mobile application security management lifecycle and security recommendations from development to completion—including usage and off-the-shelf phases.

Section 4 outlines the proposed CSA mobile application security testing scheme.


Is this
mobile application
secure?

2 Problem Statement

2.1 Mobile Computing and Application Security Challenges

Smartphones and tablets are often used for practical purposes, such as sending/receiving emails; presentation browsing; remote access of information; and even remote access to other network equipment. Mobile phones today are not only devices for receiving or making calls, but can also be electronic payment wallets, electronic identity devices and even business data storage devices. As mobile phone performance becomes more advanced, these devices are gradually replacing desktop computers in both the workplace and home [9, 6].

Although this functionality can improve productivity and efficiency, it's also encouraged the arrival of new security threats [9, 7]. With such functional diversity, one can imagine the severity if mobile devices are lost, stolen or come under malicious attacks. Moreover, many mobile-device users download third-party applications without looking through the fine print of the terms and conditions. Once downloaded, users may unknowingly authorize developers to access their profile information or other sensitive personal information. Privacy risks associated with individual use are often associated with various malicious malware attacks.

Mobile devices contain three major components: an operating system, hardware and applications. Mobile operating systems available in the market today are mainly iOS-, Android- or Windows-based. In an effort to identify potential security blind spots in the industry, a simple examination of the mobile device production process is helpful: Operating systems and hardware are pre-assembled by manufacturers in controlled environments, which inherently limits security concerns with those elements. Only after the phone is released to the marketplace and

installed with third-party applications by users do critical security issues arise.

With reference to the CSA Top Threats to Mobile Computing Report [2], this chapter discusses several main security challenges that mobile users are facing.

2.2 Third-Party Application-Derived Security Challenges

Increasingly, mobile phones are used for business more than entertainment. As requirements for applications become more complex, security challenges have also evolved from mainly virus-related issues to information theft and leakage. By design, security has to be considered early in the development stage.

Apart from Native Development Kits (NDKs)/Software Development Kits (SDKs), which are built-in, developers will have to refer to libraries or application program interfaces (APIs) [8], which are written by third parties. However, the security of these related libraries or APIs is often unverifiable when the development process begins [7, 2]. As such, code vetting at the testing phase will be critical in identifying security issues brought about by these libraries or APIs.

2.3 Mobile Application Development Management Challenges

In general, the mobile application development lifecycle [4] includes requirement-gathering, design, implementation, testing, quality assurance, product release and version revision. Additional attention needs to be given to the following development management principles:

Environment management

Mobile application development must take into consideration that the finished product will have to be installed to different versions of operating systems and configurations of mobile terminal devices and/or tablets. Otherwise, issues with incompatibility or security information leakage may result.

Version management

Similar to traditional software development, application development is usually the result of teamwork. There are many concerns a development team needs to consider, including connection scheme, offline processing, data overwriting and surrounding detection [5, 9]. Therefore, ensuring that every part of the development process is well-synchronized and documented is important to the security of mobile applications.

Native development kit management

Many application development teams implement already constructed kit components when creating certain functions. In most cases, application engineers download kits from other developers without verifying the security of the kits [6, 7, 9]. As such, native development kits should be

managed and monitored to avoid security problems.

Quality assurance management [4]
Quality assurance has to be included in any development lifecycle. Mobile application development is more complicated than traditional software development and the stability in developing iOS or Android programs are uncertain due to diversity in mobile operating systems, languages and hardware. In addition, concerns such as system security breaches in the development environment must be taken into account.

2.4 Mobile Application Security Vetting Concerns

Security problems that emerge during the development stage are mainly attributed to a lack of control in the development environment and poorly managed system calls. Key areas of concern that should be developed into vetting criteria include intentional misconduct, negligence, and native problems.

2.4.1 Intentional Misconduct

Intentional Structured Query Language (SQL) injection/Advanced Persistent Threat (APT)/Bot
This can refer to malicious actions, such as intentional injection of a virus and code, or the creation of a backdoor. Put simply, it is the performance of attacks or theft activities when implementing software or an application to achieve specific malicious goals [6, 7, 8].

Injection of malicious channel leakage and vulnerabilities [8, 7]
This can refer to the injection of malicious channel leakages and vulnerabilities into applications. This is undetectable by users and causes subsequent security problems.

API/Library (LIB) fraud/fake
When the source of API/LIB packages and versions are not managed and synchronized properly, the application may have already been infected during the development phase due to the use of fraud/fake API/LIB [9, 8]. Although such use of fraud/fake API/LIB can be unintentional, it can still lead to serious consequences during application development.

2.4.2 Negligence

Usage of obsolete or deprecated function
Developers might often use obsolete API/LIB while developing an application. For example, when a developer is creating an application using Android 2.2 API/LIB in an Android 5.0 environment [7, 9], unnecessary resource wastage or the creation of security vulnerabilities may happen (although this will not impose security risks immediately).

Usage of poorly written and non-validated code from the community
Developers may choose to seek help from the Internet to develop part of

their code. As a result, it is possible that poorly written code, that is incomplete or contains security vulnerabilities, may be used [6, 8]. This may lead to a mobile application that doesn't perform to its specifications or may introduce security concerns [2].

Usage of API/LIB

Misusing API/LIB can result in a zero-day security problem. For example, when developing an Android application, call of Apache.jar will be required [6]. It will not be a concern if the developer has downloaded a legitimate Apache.jar for the application. However, there are many modified Apache.jar configurations on the Internet and if the incorrect version is used, a zero-day scenario may result.

2.4.3 Native Problems

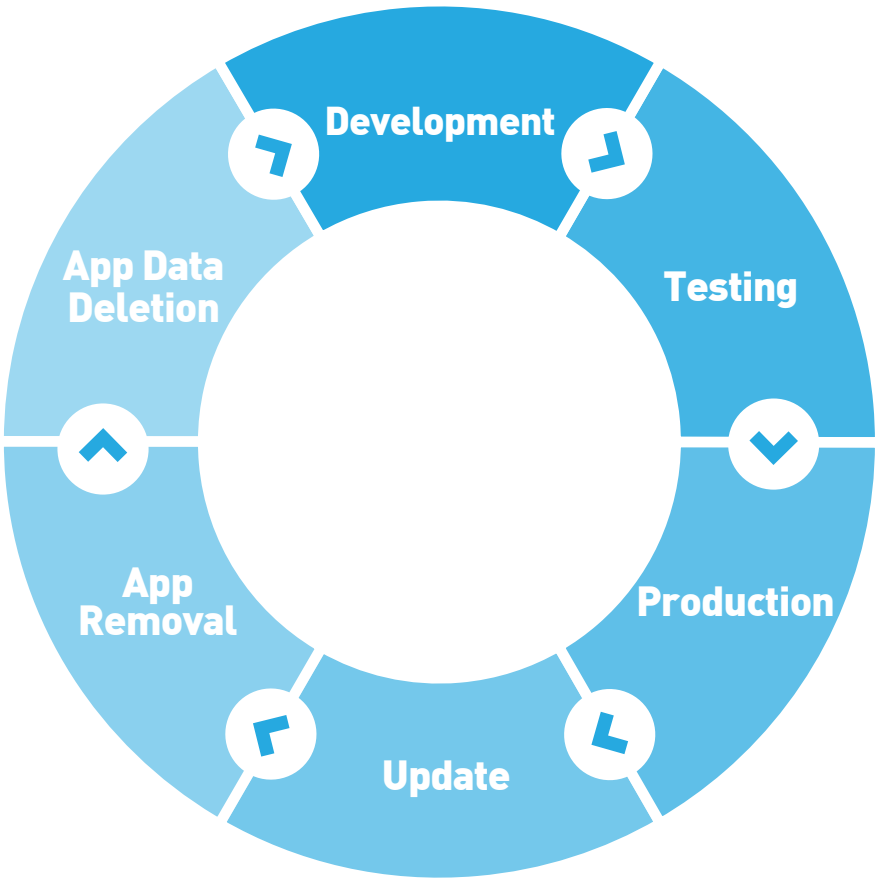
Application Native Development Kit (NDK)/Software Development Kit (SDK) native problem

While this problem is less common in practice, it refers to a situation which NDK/SDK published by the operating system manufacturer has an existing fault or backdoor [7, 8], thereby causing the application to have security vulnerabilities.

How can we manage the security of mobile applications?

3 Mobile Application Security Management Lifecycle

This section discusses the mobile application security management lifecycle and security recommendations from development to completion—including usage and off-the-shelf phases. These stages include development, testing, production, updates, application removal and data deletion [1, 4].



Mobile Application Security Management Lifecycle

Development

Apart from the project management process—which will normally focus on programming—developers should also focus on establishing security management mechanisms for security applications. This should include kits version management, code source verification and security testing.

The security recommendations for this stage are:

- Check whether kit version management is conducted as documented.
- Check whether program code origin assurance has been done by designated personnel.
- Check whether continuous security vetting management is conducted during various phases within the application development lifecycle.

Testing

Apart from security vetting that utilizes technical security measures, developers should also perform simulation and verification testing on items before pushing the application to market.

The security recommendations for this stage are:

- Check whether a standardized security vetting guideline is applied.
- Check whether vetting result feedback has been delivered to the system for future development revisions or other testing processes.

Production

This may refer to an application's public release (via venues such as Google Play or the Apple App store) or in-house release. This stage focuses on managing the submission, maintenance and operation of the release. Statistics and information such as download counts and user feedback should be periodically reviewed. Assurance management for this stage should focus on the usage of the application.

The security recommendations for this stage are:

- Perform version control during application production.
- Make sure application specifications follow the rules given by public markets such as the Apple App store and Google Play.
- Check whether assurance management of version control and content procedures are in place for in-house applications.

Update

The application update refers to new additions or modifications to an existing mobile application to improve service quality. The timely management of an application update reflects the security of the application.

The security recommendations for this stage are:

- Check whether an internal revision process has been established.
- Check whether all revisions have fulfilled the update requirements established by public markets, such as the App Store and Google Play.

Application removal

Application terms and conditions should be followed when applications are uninstalled from mobile devices. These applications should have their services—such as advertisement identification and future online payments—canceled in order to avoid unnecessary complications.

The security recommendations for this stage are:

- Verify whether additional services, such as advertisement identification and any future online payments, have been canceled.
- Make sure that these activities are recorded for future assurance.

Application data deletion

If an application must be uninstalled, it should be developed such that all relevant information associated with that particular application can be removed simultaneously. While most applications exhibit this ability, the danger remains that not all information will be completely erased.

The security recommendations for this stage are:

- Check whether the data is completely erased from the device after an application has been uninstalled.
- Check whether there is a mechanism that will inform the cloud administrator about any application data that may remain.

How can we test the security of mobile applications?

4 CSA Mobile Application Security Testing Scheme

4.1 Types of Security Vetting

Mobile application security vetting is conducted based on key concerns mentioned in the previous sub-section. There are two mobile application security vetting schemes this initiative will propose: vetting with source code available and vetting without source code available. It is commonly recognized that the provision of application source codes is adequate for vetting purposes. However, in practice, application source codes are not available because the party that requests security vetting may not be the developer of the application. Therefore, vetting without source code will become an alternative in this instance because it provides better vetting coverage.

Vetting with source code available

Mobile application security vetting is conducted either by the use of code review tools or by manual-source code reviews. This process needs to be in accordance with ISO 17025 standards, and the items that require vetting should be checked against items listed in said standards. Since this document provides only descriptions of the security items and their ideal vetting outcomes, the vetting process is conducted without specifications—using static or dynamic methods or a combination of both—while fulfilling the vetting requirements mentioned in this document.

- **Consistency:** Mobile vetting results should conform with results mentioned in this document. Ambiguous answers which map to two or more security items are not permitted by the vetting process and the result will be rendered invalid [3].

- **Comparability:** Mobile vetting results should agree with other existing known vetting schemes without discrepancy. Mobile vetting results should conform with results mentioned in this document [5, 9].

- **Repeatability:** When a mobile application is vetted repeatedly—both manually and using tools—the vetting results should stay the same each time and should not have discrepancies. The process should be repeated three times or more. Additionally, the same problematic configuration should be detected on two different mobile applications without discrepancies.

Vetting without source code available

Vetting without source code available refers to the adoption of tools to conduct static and dynamic vetting tests, such as iOS application archive files (iPAs) or Android application packages (APKs). This method should comply with the consistency, comparability and repeatability requirements mentioned above. This process also needs to be in conformance with ISO 17025 regulations (not addressed in this session). Mobile vetting items should be inspected statically and dynamically according to the security items defined in this document.

4.2 Mobile Application Security Requirements

The key security requirements are as follows:

Privacy handling
- Permission misuse
- Improper information disclosure

Native problem
- API/LIB Native Risk
- Application collusion activities
- Development obfuscation concerns

Protection requirement
- Connection encryption strength
- Data storage

Execution environment
- Power consumption

The eight points identified above are mainly developed from OWASP and NIST SP 800-163 [8, 9]. These classifications primarily focus on defining the vulnerabilities generated by processes like data protection, program coding, library calls, environmental setups and subsequent execution results during application development. Regardless of whether these vulnerabilities are created intentionally or due to negligence, a significant loss may still result from the use of mobile applications. With reference to the CSA Security Guidance for Critical Areas of Mobile Computing Section 6.2, several practices are recommended to developers to ensure the security and safety of their applications [1].

4.2.1 Privacy Handling – Permission Misuse

A mobile application will always request for certain permissions, including the retrieval of user passwords or information needed to deliver services. Application developers should evaluate whether the permission is granted excessively or is indeed required. For example, permission to access all passwords on a mobile device is unnecessary for an application that exclusively provides publicized information, such as city bus schedules. If such a request is embedded in the application, this should be identified as a type of permission misuse.

Improper permission requests for malicious purposes

- Evaluate if any permission request is malicious or does not relate to the services provided by the mobile application [6, 7, 9].

Intended hidden permission usage

- Evaluate if a developer intentionally hides the usage of permission from users. For example, users are not notified through messages or popups when the developer is declaring permission [7, 8].

Custom-built permission

- Evaluate if a developer uses permissions that are not allowed by NDK/SDKs. These custom-built permissions can be implemented using destructive or malicious third-party libraries.

4.2.2 Privacy Handling – Improper Information Disclosure

Potential private- or sensitive-data leakage or exposure may happen during mobile application usage if strict development control or vetting mechanisms are absent in the application development phase. Developers may maliciously or negligently transfer geolocation or related information to improper destinations or other applications because they may not have considered the security of such information during the application development process. Such actions are improper information disclosure and most of these disclosures are the results of poor software development practices. The following recommendations can be used to evaluate the security of application development:

Improper surrounding information disclosure

- Determine if a mobile application is programmed in such a way that information—such as an NFC physical address, MAC address, Bluetooth data and GPS location— may be leaked or sent to an unknown destination [7, 9].

- Ascertain if a mobile application is forging information—such as an NFC physical address, MAC address, Bluetooth data and GPS location—for unknown usage.

Application internal activities

- Evaluate if an application has protection mechanisms. This system

should focus on the development of Android components such as Activity, Service, BroadcastReceiver and Content Provider, or iOS components such as Container View. Effective implementation of these mechanisms will prevent private or sensitive information from unintended disclosure.

- Evaluate if an application has a protection mechanism on attributions generated by Android components such as Activities, Service and Content Provider or iOS components such as Container View. This should be established during the execution phase. Such protection mechanisms should prevent leakage or transmission of the attributions to an unknown destination.

- Evaluate if an application is programmed in such a way that will collect private or sensitive information which will be leaked or sent to an unknown address or destination using reverse engineering or malicious code.

4.2.3 Native Security – API/LIB Native Risk

Development kits such as API, LIB and SDK are often used in mobile application development to help deliver certain functionalities. Developers may use blacklisted or malicious kits to develop mobile applications out of malice or negligence. These kits may introduce risks to mobile applications as the security of the kits may not have been tested. As such, the use of these kits may introduce native risks that are related to API/LIB [7, 8]. Native risks include the following:

Potential API risks

- Identify if the application calls or uses an API that is known to be risky by the development community.
- Identify if the API calling method is associated with flaws that make the application vulnerable to attacks, introducing information leakage as a result.
- Identify if the API has native risks associated with malicious descriptive commands.

Potential LIB risks

- Identify if the application calls or uses an LIB that is known to be risky by the development community.
- Identify if the LIB calling method is associated with flaws that make the application vulnerable to attacks, thus introducing information leakage.
- Identify if the LIB has native risks associated with malicious functions.

Injection risks

- Identify if the mobile application is vulnerable to SQL Injection.
- Identify if the mobile application is vulnerable to Command Injection.

4.2.4 Native Security – Application Collusion Activity

After a mobile application is installed and executed on a mobile device, it will communicate with other applications by exchanging generated information, or will function in a permanent mode that runs in the background of the mobile device. Untraceable data leakage may happen since these applications collude with one another by exchanging information. In some cases, these hidden data leakage channels are already maliciously and systematically embedded in the application through methodologies shown below:

Data source/destination collusion

- Alert other applications about the source and storage destination of the application execution-related information.
- Retrieve data from other applications data zone during execution.

BroadcastReceiver components or equivalent

- Exchange data through BroadcastReceiver components or equivalent which are normally used for receiving important messages [8].

Data creation/modification/deletion

- Follow commands or requests from other applications to create/ modify/delete database or file resources [8].
- Send requests or commands to request other applications to create/ modify/delete database or file resources [8, 9].

4.2.5 Native Security – Development Obfuscation Concern

Irrelevant functionalities or unintended mechanisms can be created during debugging or exceptions handling. The following are three major problems that are associated with development obfuscation:

Native code execution obfuscation

- Insertion of a specific function into the application source code so that the native code will perform an unintended action during its execution.

Call mapping issues

- Call mappings are always embedded with functions that can confuse the control flow and destroy the permission mechanism of a mobile application. As such, call mapping-related functions should be vetted for any security issues.

Recreational obfuscation

- Through disassembling part of a mobile application source code and reassembling the code after testing, normal sandbox protections will not be able to detect the risks associated. Developers must understand the purpose of using obfuscated code or test the result of the obfuscated code while vetting a mobile application.

4.2.6 Protection Requirement – Connection Encryption Strength

Connectivity with the cloud is required for most mobile applications. Encryption on data in transit is required for the transmission of data between mobile devices and the cloud to ensure the data is protected against any eavesdropping or stealing. Through vetting the connection between an application and cloud, it allows developers to understand the basic security level of the application. The strength of the encryption algorithm also affects the security level of an application and should be considered during the security vetting process. The following are the security requirements that are related to connection encryption:

Connection protection

- Evaluate if the connection protection mechanism has been taken into consideration during the development of an application [8].
- Evaluate if the embedded connection protection mechanism has any well-known vulnerabilities.

Cryptographic strength and multifactor authentication

- If the encryption of mobile application data in transit uses any of the algorithms below, the data is considered to be strongly cryptographic protected.
  - Advanced Encryption Standard (AES) cryptographic algorithm. Key length should be at least 128 bits [6, 7].
  - Multi-factor authentication should also be considered for protecting data in transit.

4.2.7 Protection Requirement – Data Storage

Apart from connecting to the Internet, mobile applications can also permanently or temporarily store data on a mobile device. As such, data storage protection mechanisms are often considered for privacy and security reasons. Therefore, whether data storage protection is taken into account or not, both security and privacy are the key indicators for mobile application security. The following are two data storage security factors:

Storage mechanism and location [8]

- Evaluate if encryption has been considered in the storage mechanism.
- Evaluate if there is any concern regarding the improper location disclosure of a database or file.
- Evaluate if there is any special provision done for concealing the storage location.
- Evaluate if the data is stored in an easy-to-steal or easy-to-attack location.

Private and sensitive information protection

- Evaluate if there are any special security measures undertaken with stored information which has been classified as sensitive. For example, perform encryption or data segmentation to make it hard to clearly identify said information.

4.2.8 Execution Environment – Power Consumption

Power consumption is one of the issues that has been neglected during the development and use of many mobile applications. Excessive power consumption may decrease the sustainability of a mobile device. As such, developers should consider power consumption as one of the requirements for mobile application security testing. The following security requirements are related to mobile power consumption and their vetting guidelines:

Central processing unit (CPU) utilization rate

- Evaluate if an application utilizes CPU more than average in standby mode or primarily while in operation.

Input/output (I/O) issue

- Evaluate if an application utilizes I/O more than average in standby mode or primarily while in operation.

4.2.9 Summary

The table of summary below highlights the mobile application security vetting requirements.

Privacy Handling

Permission Misuse

- Improper permission requests for malicious purposes
- Intended hidden permission usage
- Custom-built permission

Improper Information Disclosure

- Improper surrounding information disclosure
- Application internal activities

Native Security

API/LIB Native Risk

- Potential API risks
- Potential LIB risks
- Injection risks

Application Collusion Activity

- Data source/destination collusion
- BroadcastReceiver components or equivalent
- Data creation/modification/ deletion

Development Obfuscation Concern

- Native code execution obfuscation
- Call mapping issues
- Recreational obfuscation

Protection Requirement

Connection Encryption Strength

- Connection protection
- Cryptographic strength and multifactor authentication

Data Storage

- Storage mechanism and location
- Private and sensitive information protection

Execution Environment

Power Consumption

- CPU utilization rate
- I/O issue

Summary of Mobile Application Security Requirements

4.3 Mobile Application Security Vetting Methodology

4.3.1 Definition, Requirements and Objectives

Definition

Application security vetting is the use of an automated testing tool or manual inspection to detect if there are any problems associated with the application during its development stage that can potentially threaten user information security. The identification of malicious code or security risks related to developer negligence hidden in programs—such as APK and iOS iPA files—is necessary. Additional security items defined in this document should also be considered when vetting.

Requirements

There are two types of application security vetting: one with source code available and one without source code available. Please refer to Section 4.1 of this document for the specifics of each.

Objectives

To reduce the risk of application data exfiltration or data theft.

4.3.2 Content Classification and Rating

Mobile application security vetting can be classified into three levels [3, 5, 9]:

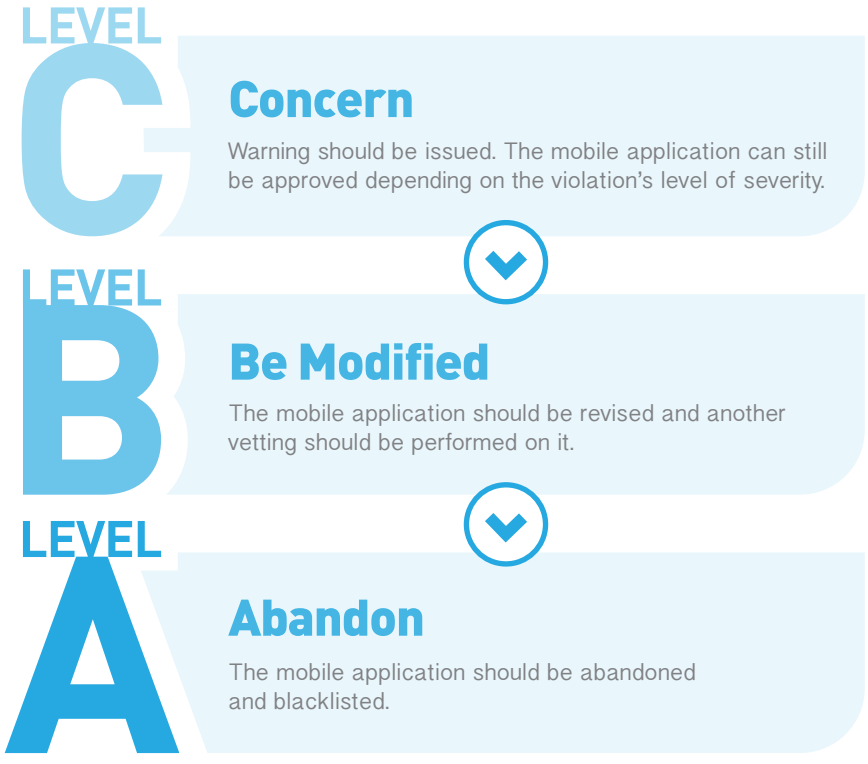
- Level A represents the different categories.
- Level B represents sub-categories.
- Level C represents security concerns.

Level A
Violations at this level mean the mobile application must be abandoned and blacklisted. Every violation that occurs at this level means that a malicious mobile application has been developed. As a result, the application will not pass the vetting process.

Level B
Warnings are issued for every vetting security requirement violation. If modifications are carried out before the application is submitted for vetting, approval is possible. However, if consecutive Level B violations occur, it will be escalated to become a Level A infraction. For example, if B1 and B2 are violated, it is re-classified as a Level A1 infringement.

Level C
At this level, warnings are given for every vetting security infraction. When consecutive Level C violations transpire, the matter will be escalated to a Level B classification. For example, if C1, C2 and C3 infractions occur, the issue is categorized as a Level B1 violation.

Mobile Application Security Testing Content Classification

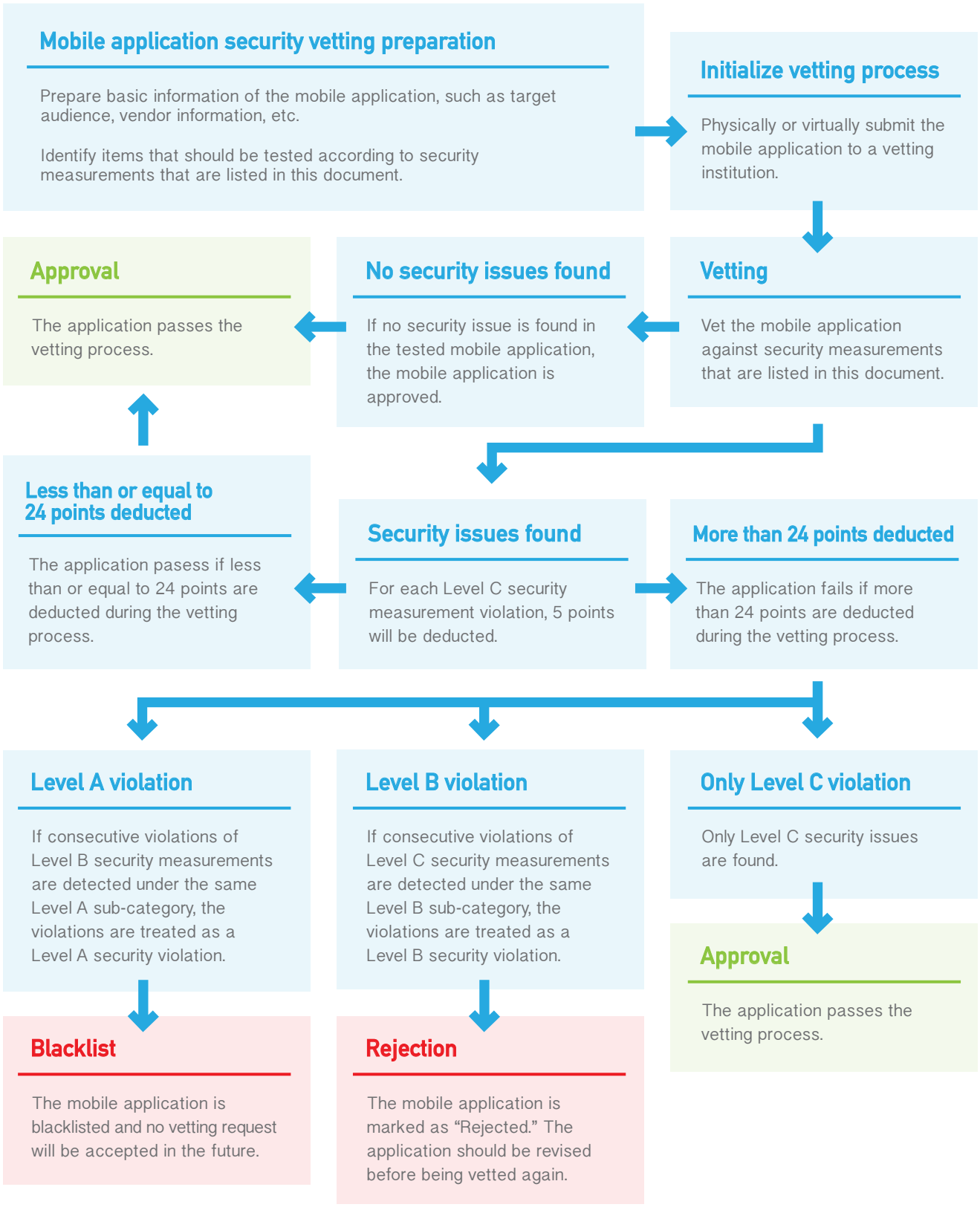


Category	Category Number	Sub-category	Sub-category Number	Security Concerns	Concern Number
Privacy Handling	A1	Permission misuse	B1	Improper permission requests for purposes	C1
				Intended hidden permission usage	C2
				Custom-built permission	C3
		Improper information disclosure	B2	Improper surrounding information disclosure	C4
				Application internal activities	C5
Native problem	A2	Application program interface (API)/ Library (LIB) native risk	B3	Potential API risks	C6
				Potential LIB risks	C7
				Injection risks	C8
		Application collusion activities	B4	Data source/destination collusion	C9
				BroadcastReceiver components or equivalent	C10
				Data creation/modification/deletion	C11
		Development obfuscation concerns	B5	Native code execution obfuscation	C12
				Call mapping issues	C13
				Recreational obfuscation	C14
Protection requirement	A3	Connection encryption strength	B6	Connection protection	C15
				Cryptographic strength and multifactor authentication	C16
		Data storage	B7	Storage mechanism and location	C17
				Private and sensitive information protection	C18
Execution environment	A4	Power consumption	B8	Central processing unit (CPU) utilization rate	C19
				Input/output (I/O) issue	C20

An example of mobile application security vetting classification scheme

4.3.3 Steps and Procedures

The followings are the steps and procedures for mobile application security vetting:



Bibliographic Citations

[1] Cloud Security Alliance, *Security Guidance for Critical Areas of Mobile Computing*, November 2012. https://downloads.cloudsecurityalliance.org/initiatives/mobile/Mobile_Guidance_v1.pdf (accessed May 15, 2016)

[2] Cloud Security Alliance, *Top Threats to Mobile Computing*, October 2012. https://downloads.cloudsecurityalliance.org/initiatives/mobile/top_threats_mobile_CSA.pdf (accessed May 15, 2016)

[3] D.R. Kuhn, R. Kacker and Y. Lei, *NIST Special Publication (SP) 800-142: Practical Combinatorial Testing*, October 2010. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-142.pdf (accessed May 15, 2016)

[4] International Organization for Standardization, *ISO/IEC 27034:2011: Information technology — Security techniques — Application security*, November 2011.

[5] K. Scarfone, M. Souppaya, A. Cody and A. Orebaugh, *NIST Special Publication (SP) 800-115: Technical Guide to Information Security Testing and Assessment*, September 2008. http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf (accessed May 15, 2016)

[6] Ministry of Industry and Information Technology of the People's Republic of China, *YD/T 2407-2013: Technical requirements for security capability of smart mobile terminal*, April 2013.

[7] Ministry of Economic Affairs (Taiwan), *Mobile Application Security Vetting Specifications*, April 2015. http://www.communications.org.tw/news/policy/item/download/31_65589e081c93691902a1e0ce75e2161e.html (accessed May 15, 2016)

[8] Open Web Application Security Project, *OWASP Mobile Security Project*, https://www.owasp.org/index.php/OWASP_Mobile_Security_Project (accessed May 15, 2016)

[9] S. Quirolgico, J Voas, T. Karygiannis, C. Michael and K. Scarfone, *NIST Special Publication (SP) 800-163: Vetting the Security of Mobile Applications*, January 2015. http://dx.doi.org/10.6028/NIST.SP.800-163 (accessed May 15, 2016)