Computer Standards & Interfaces 36 (2014) 524–530

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A standard for developing secure mobile applications

Stephen M. Dye [a,*], Karen Scarfone [b,1]

[a] Tapestry Technologies, Inc., 5000 Letterkenny Road, Chambersburg, PA 17201, United States
[b] Scarfone Cybersecurity, 13632 S. Springs Dr., Clifton, VA 20124, United States

ARTICLE INFO

Article history:
Received 3 June 2013
Received in revised form 8 September 2013
Accepted 19 September 2013
Available online 22 October 2013

Keywords:
Mobile device security
Mobile application security
Application security
Cyber security

ABSTRACT

The abundance of mobile software applications (apps) has created a security challenge. These apps are widely available across all platforms for little to no cost and are often created by small companies and less-experienced programmers. The lack of development standards and best practices exposes the mobile device to potential attacks. This article explores not only the practices that should be adopted by developers of all apps, but also those practices the enterprise user should demand of any app that resides on a mobile device that is employed for both business and private uses.

1. Introduction

The United States Department of Defense (DoD) published the first publicly available standard that mobile applications (apps) can be developed by and tested against. The long-awaited standard enables DoD personnel to more safely use a variety of apps that will improve their mission performance, as well as to take advantage of apps to perform many tasks that cannot easily be accomplished on laptops. Unfortunately, in bringing mobile devices to the DoD workplace, the threat of data disclosure or an accidental or intentional bridge between a DoD network and the public Internet is significant. Because of the DoD's high security needs and the large volume of threats against its networks, the DoD cannot allow apps to be used without thoroughly vetting their functionality and analyzing their potential vulnerabilities.

Regardless of the stringent security measures practiced for mobility by the DoD, there are numerous attack surfaces on a smartphone or tablet. These attack surfaces appear in the form of the high volume of widely used apps. Each app represents a plethora of vulnerabilities not only for the device and all data on it but also for the networks to which the device is attached. Bringing a smartphone or tablet hosting a rogue app into a DoD building has the potential to compromise a DoD network in multiple ways, allowing an attacker to gain access through the many avenues a badly written or maliciously developed app offers.

To combat this, the DoD's Defense Information Service Agency (DISA) developed a standard that may be used not only for developing new apps but also for testing, vetting, and assessing existing apps. This will provide a considerable degree of protection through applying controls and best practices in use throughout the industry to reduce vulnerabilities. The standard, known as the Mobile Applications Security Requirements Guide [1] or the SRG, is available for public download from the Information Assurance Support Environment.

This paper discusses key technical highlights from the standard. Section 2 explores app vulnerabilities while Section 3 focuses on operating system (OS) vulnerabilities. Section 4 discusses cryptography concerns. Section 5 covers the security concerns not addressed in Sections 2, 3, and 4. Section 6 examines mobile device software testing and Section 7 briefly discusses related work in mobile security. Finally, Section 8 provides a conclusion for the paper.

2. App vulnerabilities

One of the most common sources of vulnerabilities for mobile devices is the actual vulnerabilities in the apps themselves. This section discusses these vulnerabilities in the following categories: the app code, input handling, initialization, termination, and external code.

2.1. The app code

The app code itself is the primary source of most app vulnerabilities. The DoD requires that all parameters be initialized upon app startup to prevent any values that would potentially cause a frozen or unstable condition for the app, making the device more vulnerable and easier to exploit. Controlling code is also of great importance, so any source code that is never executed during runtime must not be included in an app unless the source code was provided by an approved third party.

* Corresponding author. Tel.: +1 703 585 9399.
E-mail addresses: sdye@tapestrytech.com (S.M. Dye), karen@scarfonecybersecurity.com (K. Scarfone).
[1] Tel.: +1 703 401 1018.

0920-5489/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.09.005

App code must never include hardcoded references to external resources (e.g., an external IP address). Apps must not call functions that are vulnerable to buffer overflows. Race conditions in which the app becomes unstable must also be avoided. Finally, the app must not contain known malware; this may seem obvious, but the rate of app developers using freeware routines and libraries that contain known malware remains high.

2.2. Input handling

A key distinguishing feature of apps relative to traditional desktop applications is that apps often have a much broader set of inputs. In traditional applications, the input usually consists of standard keyboard characters. In mobile apps, user input may also consist of swiping or tapping fingers on the display. Apps may also accept inputs from sensors on the device (e.g., a GPS radio, accelerometer, gyroscope, and ambient light sensor.) Ideally, all forms of input are considered in an information assurance (IA) assessment. However, the SRG currently considers only character inputs.

Traditionally, the inputs to a program are where weaknesses can be exploited, and an app is equally susceptible. When an app requires user input, the input character set must be defined and constrained. This means if the user is required to enter only numbers, then the app must define the input set to be in the range 0–9, rejecting all letters and symbols. An app that accepts undefined characters may experience unanticipated behaviors. Equally important, the input field must be designed to not be vulnerable to XML and SQL injection attacks. Injection attacks may result in an immediate loss of integrity of the data. If an app does not permit injection, then the risk of exploits from this form of attack is greatly reduced.

Format string vulnerabilities usually occur when invalidated input is entered and is directly written into the format string used to format data in the print style family of C/C++ functions. Format string vulnerabilities may lead to information disclosure vulnerabilities and may also be used to execute arbitrary code, so the SRG requires that apps must not be vulnerable to such conditions. If an attacker can manipulate a format string, this may result in a buffer overflow. If the app code does not contain format string vulnerabilities, then the risk of buffer overflows and other software exploits is significantly mitigated.

2.3. Initialization

The behavior of an app must be controlled within limits to prevent exploitation by a third party when the app fails to initialize. If an app relies on external security functions such as software modules that encrypt data, then the app must shut down, reset, or perform some safeguard action if a security module or function is unavailable. This requirement applies at app startup and during runtime. While mobile apps primarily rely on mobile OS security controls, a mobile app may contain security functions that enable the device and user to operate in a secure manner.

For example, the mobile app may operate its own cryptographic modules for data at rest and data in transit. If these modules are not present, then all data, the device, and the network would be at risk to exposure and intrusion from an unauthorized user if the app does not prevent its own execution. This measure mitigates risk and exposure from being compromised due to failed or disabled security modules. When the app shuts down it must cease running and not just deny services to a user. Other response actions might include writing an entry to the audit log, notifying the user, or limiting access to particular app features, such as the ability to export data.

2.4. Termination

For many mobile apps, the only state that is known to be compliant is the initial state, because there is no documented security policy regarding state transitions. An app could be compromised, providing an attack vector to the app and OS if shutdown and aborts are not designed to keep the app in a secure state. DoD therefore requires that an app fails to an initial state if it unexpectedly terminates, thus returning the app and device to the secure state they were previously in. An app maintains a secure state when there is strong assurance that each of its state transitions is consistent with the app's security policy. If the app fails without closing or shutting down processes or open sessions, authentication and validation mechanisms do not provide sufficient protection against unauthorized access to the app and all stored data. Securing the app to its initial level of security in the event the app crashes or terminates will mitigate the threat of an unauthorized user taking control of the device and accessing the app and stored data, compromising integrity and confidentiality.

The SRG requires that upon termination, each app remove all temporary files and tracking cookies it created during the session. Temporary files left on the system after an app has terminated may contain sensitive information, including authentication credentials or session identifiers that would enable an adversary to re-launch the app, gain unauthorized access to resources, or otherwise breach the confidentiality or integrity of the data stored on the device. Removing such files when an app terminates greatly mitigates the risk of this type of attack. Finally, any memory blocks that were used to store and process sensitive data must be cleared or overwritten to completely eradicate any trace of that data. Unless an app does this, the possibility exists for an attacker to crash the app, then analyze a memory dump of the app for sensitive information. Clearing memory will ensure that the app can operate more securely, with greater protection applied to sensitive data that will be properly removed when no longer required.

Finally, in the area of app termination, the SRG also covers transaction-based issues. Transaction-based systems must have transaction rollback, journaling, or technical equivalents implemented to ensure that the system can recover from an attack or faulty transaction data, preventing denial of service attacks.

2.5. External code

Mobile code is code downloaded from a remote source and executed on the device without user direction. Typically, mobile code is executed within web browsers. However, some apps may have the capability to execute such code in a similar fashion to a web browser. This poses a significant IA risk because such code would not have been reviewed and could perform unauthorized functions. The SRG requires that all apps must comply with the DoD Mobile Code Policy which essentially means the app must validate the signature on all ActiveX and script languages interpreted at the OS command level. This is also true for code that has full functionality to the services and resources of a device such as Java mobile code and the various scripting languages running within the confines of a browser. DoD is assuming that a level of security exists through acknowledging a valid digital signature, which implies that a trusted source created the code, and that it contains no malware. If no signature is present or the signature could not be verified, then the app must not execute it. Further to this, DoD requires that any mobile code in the app not only be signed, but also be mobile code that has already been categorized. Any uncategorized code, even though potentially safe, must not be used.

Embedding interpreters in an app to invoke prohibited code will expose the device and stored data to many forms of malicious attack. Prohibited code is intentionally not used in order to maintain the security and integrity of the device and all stored data. The SRG therefore requires that the app code must not include interpreters for any prohibited code as well.

Before external code that the app downloads during runtime may be executed, the DoD requires the user to (i) be notified that the code is being downloaded and (ii) be given the option to authorize execution of this code. Finally, the DoD requests that any APIs or external