Author(s): Luis Fernando Obando Velez (JhetoXekri)
Contact us:jheto.xekri@outlook.com

Date: 19/04/2015
Product:Parse.com SDK
Vendor: Notified 19/04/2015

We have discovered that some products from "Parse "company (http://www.parse.com/), presents a big hole regarding aninsecure keys storage. The basis of this exploit is the way in which **PARSE** products works and how it store a secure key and client ID used by many applications, if an application developed with Parse.com SDK is decompiled to extract confidential keys.

UNSECURE KEYS STORAGE

Initial Summary

This report contains a summary of the security flaws identified in the application using reversing methods dynamic and/or manual security analysis techniques. This is useful for understanding the overall security quality of an individual application or for comparisons between applications. (In this case is only an initial audit)

Application Business Criticality: (Very High)

Impacts: Operational Risk (High), Financial Loss (High) An application's business criticality is determined by business risk factors such as: reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations. The Audit Level and required assessment techniques are selected based on international standards and best practices of the **IC IACS.**

About My Methodology

Iuse static and static and reversing analysis (for mobile applications) or manually to inspect apk and binaries and identify security flaws in your applications. Using both static and reversing or manually analysis helps reduce false negatives and detect a broader range of security flaws. The static binary analysis engine models the binary executable into an intermediate representation, which is then verified for security flaws using a set of automated security scans. Once the automated process is complete, a security technician verifies the output to ensure the lowest false positive rates in the industry. The end result is an accurate list of security flaws for the classes of automated scans applied to the application.

My Rating System Using Multiple Analysis Techniques

"Think like an Attacker"
Higher assurance applications require more comprehensive analysis to accurately score their security quality. Because each analysis technique (automated static, automated dynamic, manual penetration testing or manual review) has differing false negative (FN) rates for different types of security flaws, any single analysis technique or even combination of techniques is bound to produce a certain level of false negatives. Some false negatives are acceptable for lower business critical applications, so a less expensive analysis using only one or two analysis techniques is acceptable. At higher business criticality the FN rate should be close to zero, so multiple analysis techniques are recommended.

Application Security Policies

My service allows an organization to define and enforce a uniform application security policy across all applications in itsportfolio. The elements of an application security policy include the target Level for the application; types of flaws that should not be in the application (which may be defined by flaw severity, flaw category, CWE, or a common standard including OWASP,CWE/SANS Top 25 or PCI).

Business Criticality

The foundation of rating system is the concept that more critical applications require higher security quality scores to be acceptable risks. Less business critical applications can tolerate lower security quality. The business criticality is dictated by the typical deployed environment and the value of data used by the application. Factors that determine business criticality are: reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations.US. Govt. OMB Memorandum M-04-04; NIST FIPS Pub. 199 (Business Criticality Descriptions) and others internationals requirements.

Business Criticality Definitions

Very High: This is typically an application where the safety of life or limb is dependent on the system; it is mission critical the application maintain 100% availability for the long term viability of the project or business. Examples are control software for industry, video games, commerce, transportation or governmentapplications or critical business systems such as financial trading systems.

Impact of the Vulnerability Discovered

Confidentiality Impact
According to CVSS, this metric measure the impact on confidentiality if an exploit should occur using the vulnerability on the target system. At the weakness level, the scope of the Confidentiality in this model is within an application and is measured at three levels of impact -None, Partial and Complete.

Integrity Impact
This metric measures the potential impact on integrity of the application being analyzed. Integrity refers to the trustworthiness and guaranteed veracity of information within the application. Integrity measures are meant to protect data from unauthorized modification. When the integrity of a system is sound, it is fully proof from unauthorized modification of its contents.

Availability Impact (The most important attribute for ICS IACS Industries)
This metric measures the potential impact on availability if a successful exploit of the vulnerability is carried out on a target application. Availability refers to the accessibility of information resources. Almost exclusive to this domain are denial-of service vulnerabilities. Attacks that compromise authentication and authorization for application access, application memory, and administrative privileges are examples of impact on the availability of an application.

Understand Severity, Exploitability, and Remediation Effort

Severity and exploitability are two different measures of the seriousness of a flaw. Severity is defined in terms of the potential impact to confidentiality, integrity, and availability of the application as defined in the CVSS, and exploitability is defined in terms of the likelihood or ease with which a flaw can be exploited. A high severity flaw with a high likelihood of being exploited by an attacker is potentially more dangerous than a high severity flaw with a low likelihood of being exploited. Remediation effort, also called Complexity of Fix, is a measure of the likely effort required to fix a flaw. Together with severity, the remediation effort is used to give Fix First guidance to the developer.

Flaw Severities Information

Severity	Description
Very High	The offending line or lines of code is a very serious weakness and is an easy target for an attacker. The code should be modified immediately to avoid potential attacks.
High	The offending line or lines of code have significant weakness, and the code should be modified immediately to avoid potential attacks.
Medium	A weakness of average severity. These should be fixed in high assurance software. A fix for this weakness should be considered after fixing the very high and high for medium assurance software.
Low	This is a low priority weakness that will have a small impact on the security of the software. Fixing should be consideration for high assurance software. Medium and low assurance software can ignore these flaws.
Very Low	Minor problems that some high assurance software may want to be aware of. These flaws can be safely ignored in medium and low assurance software.
Informational	Issues that have no impact on the security quality of the application but which may be of interest to the reviewer.

Policy Standards Result

- ✓ **CERT Secure Coding - Not Passed**
- ✓ **OWASP- Not Passed**
- ✓ **SANS- Not Passed**
- ✓ **ISA 62443 Not Passed**

(Preliminary Security Code Audit)

Summary of Vulnerabilities discovered

- ✓ Insecure keys storage

You can download the source code, disabled code and tutorial if you wish.

http://goo.gl/5Fz7d1

Scored: VERY HIGH

ApplicationsAffected (*)

Application:**All developed with a Parse.com SDK**

Version:**1.9.1**

Product Affected:**Parse SDK**

How to exploit it

1-Compile theVulnerableAppas demonstration (VulnerableApp.apk).

2-Decompile the apk file with apktool and dex2jar, and extract source code with JD-GUI from the jar.

3-Check the strings.xml file from the resources and ParseApplication.java from the java decompiled classes.

4- Copy the SECURE_KEY and CLIENT_ID values.

5- Create a new Android Project and add the Parse.com SDK in the libs directory.

6- Configure the new project with the cloned keys.

Know Affected Customers

Carrefour, Samsung, Vevo, MTV, GUCCI, FOX International, Barclays, AMCTV, NBA, Groupon, Cisco, McDonald's, Giorgio Armani, Whitehouse.gov and more …

Link: https://www.parse.com/customers

Know Affected Languages / Platforms

Android, iOS/OSX, .NET, JavaScript, Arduino, Embedded C and Unity.

Vector Attack:
Real world examples:

1. Attacker creates a new application with the cloned keys from any application, the users install this and the fake app, start to dump/query/modify/delete data in the official database application.

2. Attacker creates a new application with cloned keys from any application; install this in self-phone and immediacy can dump/query/modify/delete data in the official database application.

3. Attacker can develop more fake applications to uses entire sdk functions to dump/query/modify/delete/ in more customers of Parse.com (see the sdk documentation for a complete knowledge about the problem in this link https://parse.com/docs/android_guide).

Terms of Use

Use and distribution of this report is strictly confidentially, but it important to take in mind that we have to protect the entire critical infrastructure.

Trust in http://about.me/jheto.xekri If you think about this, I can develop.