TCS APP PATROL

App Name :

App Platform :

Email:

CONFIDENTIALITY STATEMENT AND DISCLAIMER

The Mobile Security tool App Patrol has a comprehensive knowledge of vulnerabilities which includes intelligence combined from OWASP Reference, Recent Threats and O.S. Platforms vulnerabilities. It should be understood that no amount of security assessment, in no mode and depth can expose all the innate security vulnerabilities. These types of assessments are intended to find the fundamental security issues within the assessment timeline. Thus it is highly recommended that even if select instances of issues are reported, the entire application be reviewed for these types of issues. App Patrol cannot be held liable for issues reported or not reported and for issues arising out of applying mitigation recommendations.

NOTE: The recommendation section provides developers, guidelines to mitigate vulnerabilities.

Security Review Findings and Recommendations

Insecure App Transport Security	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

For debugging and development purposes, developers enable the ATS keys and set required insecure values to avoid the exceptions and proper working of app. These keys and values should be removed before app goes to production. Example: - NSIncludesSubdomains NSExceptionAllowsInsecureHTTPLoads NSExceptionRequiresForwardSecrecy NSExceptionMinimumTLSVersion TLSv1.2 NSThirdPartyExceptionAllowsInsecureHTTPLoads NSThirdPartyExceptionRequiresForwardSecrecy NSThirdPartyExceptionMinimumTLSVersion TLSv1.2 NSRequiresCertificateTransparency

Instance(s)

Recommendations

- NSAllowsArbitraryLoads attribute should not be set to yes. Whitelist domains. Set secure values to other attributes also. - Pls refer below link for more information : https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33

Security Review Findings and Recommendations

Check_For_Declared_Url_Schemes	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

Apple provides the ability to launch an AIR App either from the browser or from another App through Custom URL schemes. It allows external sources to launch Apps without user interaction. The following delegate method must be used to validate custom URL schemes: "application:openURL:sourceApplication:annotation:". If this method is not implemented, then the deprecated "application:handleOpenURL:" method is called. This deprecated method does not provide all the parameters to validate the URL schemes, therefore it is important to override it with the delegate method "application:openURL:sourceApplication:annotation:" explicitly

Instance(s)

Recommendations

- Ensure that URL schemes input data is validated and sanitised.
Implement the following method to validate and sanitise the URL:
(BOOL)application:(UIApplication *)application openURL:(NSURL *)url
sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{ // Validate if the source App is acceptable and is a secure source // if yes, parse the URL and validate URL input data // if transactional, ask the user for authentication // if authorised, navigate appropriately } - Encrypt sensitive data before passing it as input to the URL schemes.

Security Review Findings and Recommendations

Cryptid_Check	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

In reversing iOS applications, often the first step includes removing the App Store encryption. Removing this encryption allows an attacker to get a greater understanding of the internal class structure and hoe the application binary works and allows them to get the binary in a suitable state of reverse engineering. Applications which are in-house distributed and self signed are not encrypted.

Instance(s)

Recommendations

Code in the url link below checks for the existence of
LC_ENCRYPTION_INFO, and verifies that encryption is still enabled.
http://landonf.bikemonkey.org/2009/02/index.html

Security Review Findings and Recommendations

Stack_Smash_Protection_Check	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

Stack smashing protection is an exploit mitigation technique that protects against stack overflow attacks by placing a random value known as stack canary before local variables on stack. The stack canary is checked upon return of the function. In case of an overflow, the canary is corrupted, and the App is able to detect and protect against the overflow. In order to take advantage of the stack smashing protection, the App should be compiled with the "-fstack-protector-all" flag.

Instance(s)

Recommendations

Activate stack-smashing protection by specifying "-fstack-protector-all"
complier flag.

Security Review Findings and Recommendations

Pie_Flag_Check	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

Position Independent Code (PIC) is a code that can be loaded and run from anywhere in the virtual memory, that is, it does not need to be loaded at a fixed address. An App that comprises of PIC is linked as a PIE and ensures security, by allowing the dynamic loader at run-time to slide the App binary to a random location in virtual memory. This process eliminates a vector of attack that relies on otherwise predictable load offsets. To compile an App with a PIE flag, set the flags Generate Position-Dependent Code" and Don't Create Position Independent Executable to NO, in the Build Settings of the project. When either or both of these flags are set to "YES", the App loads itself at a fixed memory address, thereby increasing the vulnerability

Instance(s)

Recommendations

- Click Target > Build Settings and Select NO against Generate Position-Dependent Code and Don't Create Position Independent Executable flags

Security Review Findings and Recommendations

Automatic Reference Counting	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

ARC protects applications from memory corruption vulnerabilities by moving the responsibility of memory management from the developer to the compiler.

Instance(s)

Recommendations

ARC can be enabled in an application within XCode by setting the compiler option â€œObjective-C Automatic Reference Countingâ€ to â€œyesâ€. By default it is marked as â€œyesâ€.

Security Review Findings and Recommendations

Third_Party_Frameworks_Check	Severity - Medium Static Analysis

OWASP Reference 2017 v1.0

Not Applicable

Compliance/Security Control Reference

Not Applicable

Abstract

Vulnerability Description

Third-party libraries are used by app developers to add functionality to apps, such as using Facebook libraries for authentication. They also enable developers of free apps to make money by linking their app to them; the Google AdMob library, for instance, might access a user's location to target the user with ads, while the Flurry analytics library might gather user information for a marketing profile.

Instance(s)

Recommendations

Security auditing must thoroughly test third-party libraries and functionality as well. This should include core iOS and Android code/libraries. Upgrading to a new version of a third-party library (or OS version) should be treated as version of your app. An updated third-party library (or new OS version) can contain new vulnerabilities or expose issues in your code. They should be tested just like you test new code for your app. On iOS, statically compile third-party libraries to avoid LD_PRELOAD attacks; in such attacks a library and its functions can be swapped out for an attackerâ€™s library with functions replaced with malicious code.

Appendix : Glossary

1. Risk Rating

Severity	Description
High	The vulnerabilities under high rating are considered to be of highest risk level. Such vulnerability should be handled with highest priority. Under specific conditions, these vulnerabilities can potentially make the system unusable and lead to serious security breaches.
Medium	Though the threat is not critical at the moment, it has the potential to become a High risk threat in the future under certain circumstances if not mitigated. Medium risk vulnerabilities require significant mitigation to lower the impact of the threat.
Low	The information found is useful to the attacker, but is not a threat in itself. Existing security controls are likely to be adequate or the risk is acceptable, but over the period this may give rise to more serious problems.
Info	The data revealed is an additional piece of information and there are no serious security implications related to it. Items listed here are not vulnerabilities, but are indicators of overall application development security practices.

Risk Rating		Impact		
		High	Medium	Low
Likelihood	High	High	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Info

2. Vulnerability Title

The vulnerability title is a short one line description of the vulnerability discovered.

3. OWASP Reference

The OWASP reference is a standardized list of vulnerability types. This aims to identify each vulnerability type with a unique reference id, which may be used to access more information regarding the vulnerability.

4. Abstract

The section describes the severity of a potential attack based on successful exploitation of the vulnerability.

5. Vulnerability Description

The description gives the overview of flaw or bug that caused the vulnerability. This is a brief explanation of the vulnerability with examples.

6. Instance(s)

The section highlights vulnerabilities exist in Application scanned by plugin.

7. Recommendation

This section provides solutions or workarounds to mitigate the risk arising from this vulnerability.

8. Severity

The severity describes the risk level of the vulnerability.

9. Privacy Risk

Risk scoring of an application is based on the data obtained during risk analysis of application. Based on the data collected during the scan, the risk score is assigned to each risk and finally arrived at the overall score. The scan looks at the possible risks that an application can possess.

10. App Risk

This section rates the application based on the vulnerabilities and instances detected. A application is rated on a scale of 100 wherein the app with score in range 0 - 30 is under Low risk, 30 - 50 is under Medium risk, 50 and above is under High risk.

11. CVSS

The Common vulnerability scoring system CVSS is a NIST standard for assessing the severity of security vulnerabilities. The CVSS score establishes a measure of how much concern a vulnerability warrants, compared to other vulnerabilities. The score is arrived at considering various vectors and applying standard formulaes. The scores range from 0 to 10. Vulnerabilities with a base score in the range 7.0-10.0 are High, those in the range 4.0-6.9 as Medium, and 0-3.9 as Low.

Reference

www.first.org/cvss/cvss-guide

12. Compliance

An app must comply with privacy and data protection laws, regulations, and policies designed to protect confidential information, such as PCI DSS, NIST, HIPAA. This section provides an overview on which compliance has been violated by the app.

PCI Reference

https://www.pcisecuritystandards.org/index.php

HIPPA Reference

http://www.hhs.gov/ocr/privacy/

NIST Reference

http://csrc.nist.gov/publications/PubsSPs.html

Document Reference

https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0CC0QFjAD&url=https%3A%2F%2Fcloudsecurityalliance.org%2Fguidance%2FCSA-ccm-v1.00.xlsx&ei=I6vhU82yEpK8ugSMzoDgCQ&usg=AFQjCNH50ajXlFyJ_Q5KMCyU16tlDJSYeg&bvm=bv.72197243,d.c2E

13. Security Threat Coverage

The following security test scenarios have been covered -

Data Confidentiality	Data Storage	Storage in keychain storage
		Storage in Preferences
		Storage in NSUserDefaults
		Storage in Database
		Storage in Flat Files
		Storage in pList Files
		Storage in Local Storage

		Storage in Media/SD Card
	Data Leakage	Availability Of Stray Files
		Applications Logs enabled
		Application StackTrace
		Application Crash Logs
		Autocomplete Enabled
		Copy Paste Enabled
		Data Masking Enabled
		Availability of Hardcoded information
		Memory Analysis
		BroadCast Theft
	Information Caching	Use of App Screenshots
		Caching of URLs/Parameters/data in Web Cache
		Offline Caching of pages
	Cookie Attributes	Status of Cookie Secure flag
		Status of Cookie HTTP only
	Broken Cryptography	Strength of Symmetric algotithms used with various modes
		Strength of Hashing Algorithms Used
		Strength of Assymetric Algorithms with Various padding
		Use of Encode/Decode
		Strength of KeyGeneration Algorithm
		Key storage and sharing
	SSL	SSL Enforced
		SSL Certificate Validity Startdate , Enddate, Host name, Subjet Name
		SSL Chain of Trust
		SSL Certificate Revocation Status
		SSL Certificate CA
		SSL Protection Space Protocol
		SSL Pinning

	Data Protection	Use of Data protection API
		Use of File Protection API
		Activity hijacking
		Service Hijacking
		Insecure Declaration of Activities, Services, Content Providers, Receivers
	Network	IP Scan
		HTTP Methods
		Parameter Tampering
		Bypass application logic
Authentication and Authorisation	Password Strength	Basic password complexity
		Case Sensitivity
		Availability of hardcoded password
	Account Lockout	Limited number of incorrect attempts
		Method of Lockout
		Susceptable to Brute Force
	Caching Login details	Remember me
		Storing on Device
	Privilege Escalation	Horizontal
		Vertical
	Authentication Methods	Two factor Authentication
		Single Sign On
	Session Management	Availability of Logout feature
		Auto logout after limited period of time
		Validation of Session token
		Session ID storage
		Session ID length
		Susceptibity to Session Hijacking
	Password Management	Account Sign up/ New Account creation
		Change password
		Forgot password

Other Protection	Binary Analysis	App Signed with Debug Key
		Code Obfuscation
		PIE Enabled
		Antidebugging Enabled
		App Backup Enabled
		App accesibility on Jailbroken/Rooted Device
	Insecure Permission	Least privilege permissions access in Manifest
		Excess Manifest Permission
Data Validation	URL Modification	URL Redirection
		URL Tampering
	Access to Third Party	Custom URL Handler - Contents
		Custom URL Handler - Activities
		Custom URL Handler - Intents
		Custom URL Handler - Broadcast Receivers
		Custom URL Schemes
	Cross Site Resource Forgery	Cross Site Resource Forgery
	Cross Site Scripting	Susceptible to DOM Based XSS
		Susceptible to Reflective XSS
	SQL Injection	Susceptible to SQL Injection
	Phishing	Susceptible to phishing attacks
	DOS	Susceptible to DOS attacks
	Format String	Susceptible to Format String Attack
	Malware Injection	Susceptable to malacious script injection using reverse engineeringtechniques
		Malicious activity launch
		Malicious service launch
		TapJacking attack
		Intent Injection
	Buffer Overflow	Susceptible to Buffer Overflow
Error/Exception Handling	Alert Messages	Generic messages in Alert Prompts
	Toast Messages	Generic Messages in Toast

	Exception Handling	Uncaught Exception
		Generic Messages in Error Prompts
	Error Messages	Handling of SQL error messages
		Handling of HTTP Error messages
		Handling of Application Error messages
User Details	Privacy Checks	Location Tracking Enabled
		Email Services Enabled
		Call Services Enabled
		SMS Services Enabled
		Device Details Accessed