TCS APP PATROL

App Name : Altoro-App

App Platform : iOS

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.

Report Summary

#	Vulnerability Name	Status
1	Insecure App Transport Security	FAIL
2	Scan For Declared URL Schemes	FAIL
3	CryptID Scan	FAIL
4	Stack Smash Protection Scan	PASS
5	Pie Flag Vulnerability Scan	PASS
6	Automatic Reference Counting	PASS
7	Third Party Frameworks Scan	PASS

Insecure App Transport Security	Severity - Medium
	Status - FAIL

OWASP Reference 2017 v1.0

M3 - Insecure Communication

Compliance/Security Control Reference

Not Applicable

Abstract

App Transport Security (ATS) is a technology that requires an app to either support best practice HTTPS security or statically declare its security limitations via a property in its Info.plist. App Transport Security (ATS) is enabled as a system default behavior. When ATS is enabled, it forces an app to connect to web services over an HTTPS connection rather than HTTP, which keeps user data secure while in transit by encrypting it.

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.You can opt-out of ATS for certain URLs in your Info.plist by using NSExceptionDomains. Within the NSExceptionDomains dictionary you can explicitly define URLs that you need exceptions for with ATS. The exceptions you can use are: NSIncludesSubdomains NSExceptionAllowsInsecureHTTPLoads NSExceptionRequiresForwardSecrecy NSExceptionMinimumTLSVersion NSThirdPartyExceptionAllowsInsecureHTTPLoads NSThirdPartyExceptionMinimumTLSVersion NSThirdPartyExceptionRequiresForwardSecrecy Each of these keys allows you to granularly disable ATS or particular ATS options on domains where you are unable to support them.

Instance(s)

NSAllowsArbitraryLoads is set to True. Expected value is False

Recommendations

- NSAllowsArbitraryLoads attribute should not be set to yes. Whitelist domains, if it is set to yes. - 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

Scan For Declared Url Schemes

Severity - Medium

Status - FAIL

OWASP Reference 2017 v1.0

M7 - Client Code Quality

Compliance/Security Control Reference

NIST SP800-53 R2 SI-2

Abstract

Custom URL schemes are not just limited to launching the other App, but additional information can be passed to other App through the URL. If not implemented properly, it may lead to successful malicious script injection, corruption of data in the App and could also leak sensitive information by returning a result to its caller upon completion. This may lead to loss of Integrity.

Vulnerability Description

Apple provides the Ability to launch an third party 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)

-altdb -althttp -altref Please ensure that the URL schemes are validated in the canOpenURL app-delegate method

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.

Cryptid Scan	Severity - Medium
	Status - FAIL

OWASP Reference 2017 v1.0

M9 - Reverse Engineering

Compliance/Security Control Reference

Not Applicable

Abstract

A lack of binary protections within a mobile app exposes the application and its owner to a large variety of technical and business risks if underlying application is insecure or exposes sensitive intellectual property. A lack of binary protection results in a mobile app that can be analyzed, reverse- engineered and modified by an adversary in rapid fashion.

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.

Instance(s)

cryptid 0 found. This application is not encrypted cryptid 0 found. This application is not encrypted cryptid 0 found. This application is not encrypted

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

Stack Smash Protection Scan	Severity - Medium
	Status - PASS

OWASP Reference 2017 v1.0

M9 - Reverse Engineering

Compliance/Security Control Reference

NIST SP800-53 R2 CM-5

Abstract

Variables whose memory is allocated on the stack need to be carefully managed so that data stored in them will not exceed the stack space that has been allocated. If a malicious security cracker is able to intentionally exceed the stack space allocated to a variable, he or she can use malformed data to actually affect program control flow in a deliberate way. This sort of security compromise is known as a 'stack-smashing attack' and it may lead to OS compromise

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 application is able to detect and protect against the overflow. In order to take advantage of the stack smashing protection, the application should be compiled with the -fstack-protector-all flag.

Instance(s)

___stack_chk_fail flag found in binary. Stack smash protection is enabled ___stack_chk_guard flag found in binary. Stack smash protection is enabled

Recommendations

iOS applications which use the stack canaries will contain _stack_chk_fail and _stack_chk_guard symbols in the binary.

Pie Flag Vulnerability Scan	Severity - Medium
	Status - PASS

OWASP Reference 2017 v1.0

M9 - Reverse Engineering

Compliance/Security Control Reference

NIST SP800-53 R2 CM-5

Abstract

Address space layout randomization (ASLR) is a memory-protection process for operating systems (OSes) that guards against buffer-overflow attacks by randomizing the location where system executables are loaded into memory.

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)

PIE flag found, helps in randomizing the application objects location in the memory to prevent remote exploitation of memory corruption vulnerabilities.

Recommendations

- the application should be compiled with the -fPIE -pie flag ("Generate Position-Dependent Code" build option in Xcode). In the latest version of the XCode, this flag is automatically checked by default.

Automatic Reference Counting	Severity - Medium
	Status - PASS

OWASP Reference 2017 v1.0

M9 - Reverse Engineering

Compliance/Security Control Reference

Not Applicable

Abstract

Automatic Reference Counting implements automatic memory management for Objective-C objects and blocks, freeing the programmer from the need to explicitly insert retains and releases. It does not provide a cycle collector; users must explicitly manage the lifetime of their objects, breaking cycles manually or with weak or unsafe references.

Vulnerability Description

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

Instance(s)

_objc_release found in the binary,

Recommendations

ARC can be enabled in an application within XCode by setting the compiler option 'Objective-C Automatic Reference Counting' to 'yes'. By default it is marked as 'yes'.

Third Party Frameworks Scan	Severity - Medium
	Status - PASS

OWASP Reference 2017 v1.0

M7 - Client Code Quality

Compliance/Security Control Reference

Not Applicable

Abstract

Third party libraries are not part of the standard SDK libraries, which may involve certain vulnerabilities

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)

No third party frameworks found.

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 attacker'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=I6vhU82yEpK8uqSMzoDgCQ&usg=AFQjCNH50ajXlFyJ_Q5KMCyU16tlDJSYeg&bvm=bv.72197243,d.c2E