August 31, 2018

ASSESSMENT REPORT:
Mobile Application Penetration Test

Contoso
Darin Allison

RHINO SECURITY LABS

ASSESSMENT INFORMATION

RHINO SECURITY LAB DETAILS

Account Executive

Austin Tippett
Account Executive
austin.tippett@rhinosecuritylabs.com
206.408.8009

Assessment Team

Hector Monsegur (Lead Consultant)
hector.monsegur@rhinosecuritylabs.com
888.944.8679 x713

Spencer Gietzen
spencer.gietzen@rhinosecuritylabs.com
888.944.8679 x719

Project Manager

David Moratti
david.moratti@rhinosecuritylabs.com
888.944.8679 x704

CLIENT DETAILS

Company Information

Contoso
1234 East Pike St.
Seattle, WA
98122

Contact Information

Darin Allison
Director of Vulnerability Management
darin.allison@contoso.com
555.555.0199

ASSESSMENT SCOPE SUMMARY

Engagement Timeframe

08/27/2018 - 08/31/2018

Engagement Scope

Contoso's iOS and Android Mobile Applications

Project ID: Contoso-MobileApp-V9.0-08-27-2018

Report Date: August 31, 2018

ENGAGEMENT OVERVIEW

Rhino Security Labs specializes in mobile application penetration techniques and practices, identifying areas for improvement. At Rhino Security Labs we specialize in manual assessments that go beyond basic automated tests to identify real attack vectors that can be used against your application.

With decades of combined experience, Rhino Security Labs is at the forefront of application security and penetration testing. With a veteran team of subject matter experts, you can be sure every resource is an authority in their field.

SERVICE DESCRIPTION

Penetration Testing is the process of simulating real-world attacks by using the same techniques as malicious hackers. For a security assessment that goes beyond a simple vulnerability scanner, you need experts in the industry.

Extensive Application Assessment

With their growing complexity and demand, mobile applications have become the target of choice for hackers. Rhino Security Labs' Application Service offerings help protect your enterprise applications' mobile services from a range of security threats.

Application security issues are not only the most common type of vulnerability, they're also growing in complexity. While the OWASP Top 10 is used as the standard for identifying application security flaws, that's just a start - many advanced vulnerabilities are not included in that list. Automated vulnerability scanners and penetration testers focused on OWASP will fall behind new threats, leaving the application exposed to unknown risks.

At Rhino Security Labs, we go far beyond the OWASP Top 10, continually pushing the boundaries of application security and detailing the way unique architectures can be abused - and how to fix them.

Manage Application Security Risk

Mobile applications are no longer just for marketing. Often the interface to an entire suite of technologies, mobile applications can tie into critical databases, vulnerable libraries, and plugins, XML and LDAP capabilities, underlying operating systems and client web browsers. It's never "just a mobile app" anymore, making them even more difficult to protect.

CAMPAIGN OBJECTIVES

Vulnerability Identification

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's applications. Our assessors attempt to exploit and gain remote unauthorized access to data and systems. The detailed results of both the vulnerability scan and the manual testing are shown in this report.

YOUR ASSESSMENT TEAM

Passionate and forward-thinking, our consultants bring decades of combined technical experience as top-tier researchers, penetration testers, application security experts, and more. Drawing from security experience in the US military, leading technology firms, defense contractors, and Fortune 50 companies, we pride ourselves on both depth and breadth of information.

David Moratti - *Technical Project Manager*

David brings a breadth of information security education and experience. David manages all Rhino Security Labs engagements, performing the penetration testing for many of the engagements himself. With a degree in information security at the University of Washington, David is uniquely able to speak to both technologists and management in language understood and applied by both parties.

Hector Monsegur - *Director of Assessment Services*

Hector Monsegur brings a unique perspective from decades of offensive experience and a desire to make an impact in client security. In working with the US Government, Mr. Monsegur identified key vulnerabilities - and potential attacks - against major federal infrastructure including the US military and NASA. In his role as a security researcher at Rhino Security Labs, he has identified countless zeroday vulnerabilities and contributed to dozens of tools and exploits. In his leadership role, his unmatched technical experience is shared to both educate other operators and guide technical research. Mr. Monsegur is a leading speaker for security organizations and conferences around the world.

Spencer Gietzen - *Penetration Tester*

Spencer Gietzen came into the cyber security field with a background in web and software development. His primary focus as a penetration tester is security relating to Amazon Web Services, mobile applications, and web applications, where he has found success in discovering multiple vulnerabilities and attack vectors through extensive research. Spencer is also the lead developer of our internal AWS post exploitation framework, Pacu.

PROCESS AND METHODOLOGY

Rhino Security Labs used a comprehensive methodology to provide a security review of Contoso's mobile application(s). This process begins with detailed scanning and research into the architecture and environment, with the performance of automated testing for known vulnerabilities. Manual exploitation of vulnerabilities follows, for the purpose of detecting security weaknesses in the application.

1

Reconnaissance

The primary goal in this process is to discover crucial data about Contoso's applications, providing the foundation for a tailored penetration test. Reconnaissance is carried out via automated scanners (such as nmap), as well as application fingerprinting and discovery.

2

Automated Testing

Rhino Security Labs used a vulnerability scanner to provide a foundation for the full manual assessment, and each finding is manually verified to ensure accuracy and remove false positives.

3

Exploration and Verification

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's applications. The detailed results of both the vulnerability scan and the manual testing are shown in the tables below.

4

Assessment Reporting

Once the engagement is complete, Rhino Security Labs delivers a detailed analysis and threat report, including remediation steps. Our consultants set an industry standard for clear and concise reports, prioritizing the highest risk vulnerabilities first. The assessment includes the following:

- Executive Summary
- Strategic Strengths and Weaknesses
- Identified Vulnerabilities and Risk Ratings
- Detailed Risk Remediation Steps
- Assets and Data Compromised During Assessment

5

Optional Remediation

As an optional addition to the standard assessment, Rhino Security Labs provides remediation retesting for all vulnerabilities listed in the report. At the conclusion of the remediation testing and request of the client, Rhino Security Labs will update the report with a new risk level determination and mark which vulnerabilities in the report were in fact remediated to warrant a new risk level.

SCOPING AND RULES OF ENGAGEMENT

While real attackers have no limits on Mobile Application Penetration engagements, we do not engage in penetration testing activities that threaten our ethics and personal privacy.

Constraints

No additional limitations were placed upon this engagement, as agreed upon with Contoso.

Assessment Scope

Rhino Security Labs compiled the following notes during the reconnaissance portion of the Mobile Application Penetration Test. These notes provide the information needed to accurately assess the application(s) and test for vulnerabilities.

Contoso Mobile Client

Platform(s)

Android

iOS

Description

The Contoso Mobile App allows users access to their accounts, with both in-depth and at-a-glance information on financial information, team-member activity, and interfacing with team members.

Sensitive Assets and Processes

Financial information, user activity, and team members names.

EXECUTIVE SUMMARY OF FINDINGS

Rhino Security Labs conducted a Mobile Application Penetration Test for Contoso. This test was performed to assess Contoso's defensive posture and provide security assistance through proactively identifying vulnerabilities, validating their severity, and providing remediation steps.

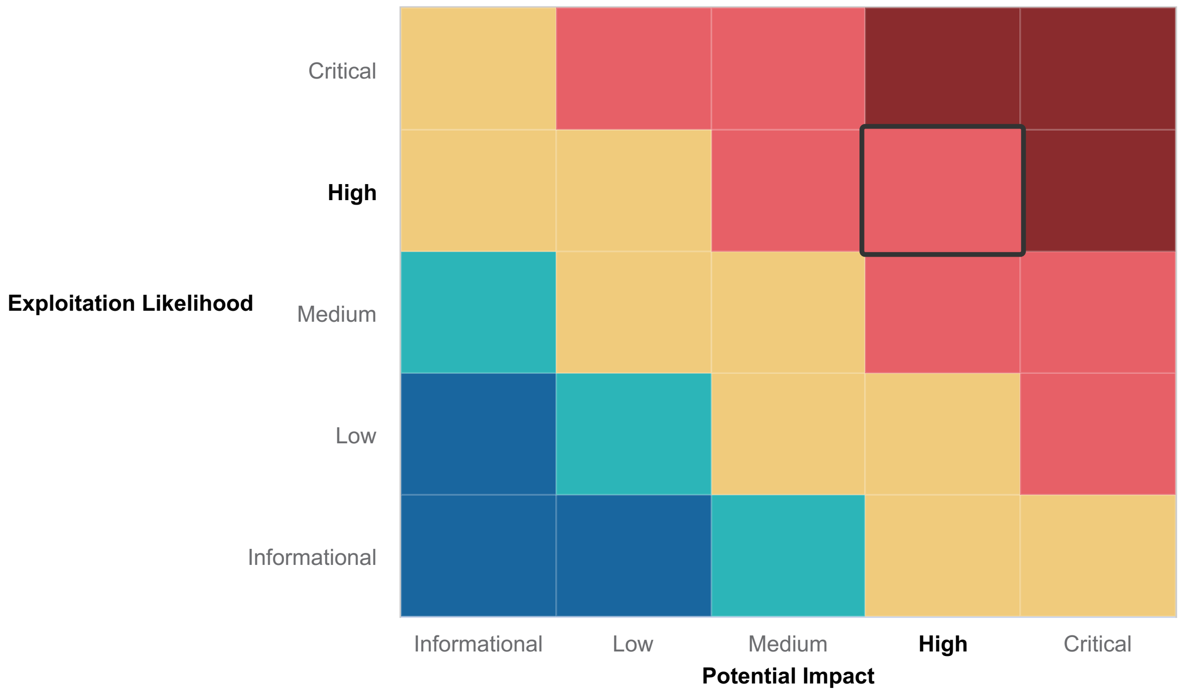
Rhino Security Labs reviewed the security of Contoso's infrastructure and has determined a High risk of compromise from external attackers, as shown by the presence of the vulnerabilities detailed in this report.

The detailed findings and remediation recommendations for these assessments may be found later in the report.

Mobile Application Risk Rating

Rhino Security Labs calculates mobile application risk based on Exploitation Likelihood (ease of exploitation) and Potential Impact (potential business Impact to the environment).

OVERALL RISK RATING: HIGH



Summary of Strengths

While Rhino Security Labs was tasked with finding issues and vulnerabilities dealing with the current environment, it is useful to know when positive findings appear. Understanding the strengths of the current environment can reinforce security best practices and provide strategy and direction toward a robust defensive posture. The following traits were identified as strengths in Contoso's environment.

1. Strong user authentication implemented.
2. Stack-smashing protection implemented on applications.

Summary of Weaknesses

Rhino Security Labs discovered and investigated many vulnerabilities during the course of its assessments for Contoso. We have categorized these vulnerabilities into general weaknesses across the current environment, and provide direction toward remediation for a more secure enterprise.

1. Sensitive information disclosure via multiple methods.
2. No jailbreak (iOS) or root (Android) detection mechanisms implemented in the applications.
3. No SSL certificate pinning implemented on iOS or Android.

Strategic Recommendations

Not all security weaknesses are technical in nature, nor can they all be remediated by security personnel. Companies often have to focus on the root security issues and resolve them at their core. These strategic steps are changes to the operational policy of the organization. Rhino Security Labs recommends the following strategic steps for improving the company's security.

1. Implement SSL certificate pinning on both the iOS and Android applications.
2. Add checks for jailbroken (iOS) or rooted (Android) devices to each application.

SUMMARY VULNERABILITY OVERVIEW

Rhino Security Labs performed a Mobile Application Penetration Test for Contoso on 08/27/2018 - 08/31/2018. This assessment utilized both commercial and proprietary tools for the initial mapping and reconnaissance of the application(s), as well as custom tools and scripts for unique vulnerabilities. During the manual analysis, assessors attempted to leverage discovered vulnerabilities and test for key security flaws, including those listed in the OWASP Top 10 Vulnerabilities list. The following vulnerabilities were determined to be of highest risk, based on several factors including asset criticality, threat likelihood, and vulnerability severity.

Vulnerability Risk Definition and Criteria

The risk ratings assigned to each vulnerability are determined by averaging several aspects of the exploit and the environment, including reputation, difficulty, and criticality.

CRITICAL	Critical vulnerabilities pose a serious threat to an organization's security, and should be fixed immediately. They may provide a total compromise of the target environment, or similar critical impacts.
HIGH	High risk vulnerabilities provide a serious risk to the company environment and should be corrected promptly. These issues can significantly affect the organization's security posture.
MEDIUM	Medium severity vulnerabilities represent a moderate risk to the environment. They may require additional context before remediation but should be remediated after critical and high risks.
LOW	Low severity vulnerabilities provide minimal risk to the target environment, and often theoretical in nature. Remediation of low risks is often a lower priority than other security hardening techniques.
INFORMATIONAL	Informational vulnerabilities have little-or-no impact to the target scope by themselves. They are included however, as they may be a risk when combined with other circumstances or technologies not currently in place. Remediation of informational items is not necessary.

VULNERABILITY SUMMARY TABLE

The following vulnerabilities were found within each risk level. It is important to know that total vulnerabilities is not a factor in determining risk level. Risk level is depends upon the severity of the vulnerabilities found.

1	3	4	2	0
Critical	High	Medium	Low	Informational

Vulnerability ID - Name And Remediation	Risk Level
C1 - SENSITIVE CACHE INFORMATION DISCLOSURE Do not store sensitive information inside the internal storage.	CRITICAL
H1 - NO SSL CERTIFICATE PINNING Implement SSL certificate pinning in all mobile applications.	HIGH
H2 - USERNAME ENUMERATION VIA PASSWORD RESET Change password recovery messages to be generic.	HIGH
H3 - IOS APP TRANSPORT SECURITY (ATS) DISABLED Ensure that the NSAllowsArbitraryLoads setting is set to false in the applications "Info.plist" file.	HIGH
M1 - SENSITIVE INFORMATION DISCLOSURE VIA SQLITE DATABASE Avoid storage of internal information where possible and encrypt the SQLite Database.	MEDIUM
M2 - NO ROOT CHECK ON ANDROID APPLICATION Implement multiple root detection checks before beginning the run time of the application.	MEDIUM
M3 - UNPROTECTED ANDROID SERVICE Explicitly set the "android:exported" property to "false" in the "AndroidManifest.xml" file for the affected services.	MEDIUM
M4 - NO JAILBREAK CHECK ON IOS APPLICATION Implement multiple jailbreak detection checks before beginning the run time of the application.	MEDIUM
L1 - APPLICATION WITH "ALLOWBACKUP" FLAG SET TO TRUE Set the allowBackup flag in the AndroidManifest.xml file to false.	LOW
L2 - ANDROID APPLICATION DEBUGGING ENABLED Disable application debugging in the "AndroidManifest.xml" file.	LOW

VULNERABILITY FINDINGS

The vulnerabilities below were identified and verified by Rhino Security Labs during the process of this Mobile Application Penetration Test for Contoso. Retesting should be planned following the remediation of these vulnerabilities.

C1 Sensitive Cache Information Disclosure

Risk Rating: Critical

Exploitation Likelihood: **High** | Potential Impact: **Critical**

Description

The mobile application may expose potentially sensitive information through local cache files. An attacker could get more information than necessary, and try to create a new attack vector over the application or the server back-end. An attack can also obtain the exact software and version running, the internal folders, etc. Additionally, in this case, an attacker could get the application code and search for sensitive data inside it.

Affected Application(s) & Platforms

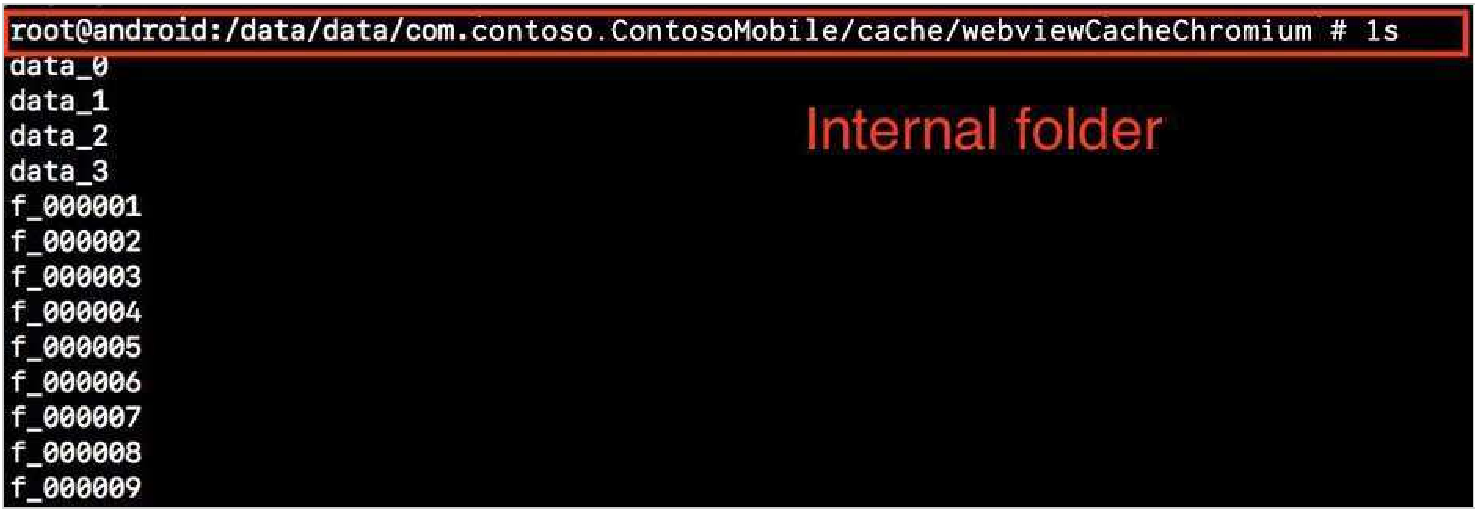
Contoso Mobile Client: Android

Remediation

We recommend to not store sensitive information inside the internal storage. Public data should be available to everyone, but sensitive and private data must be protected, or, better yet, kept out of device storage.

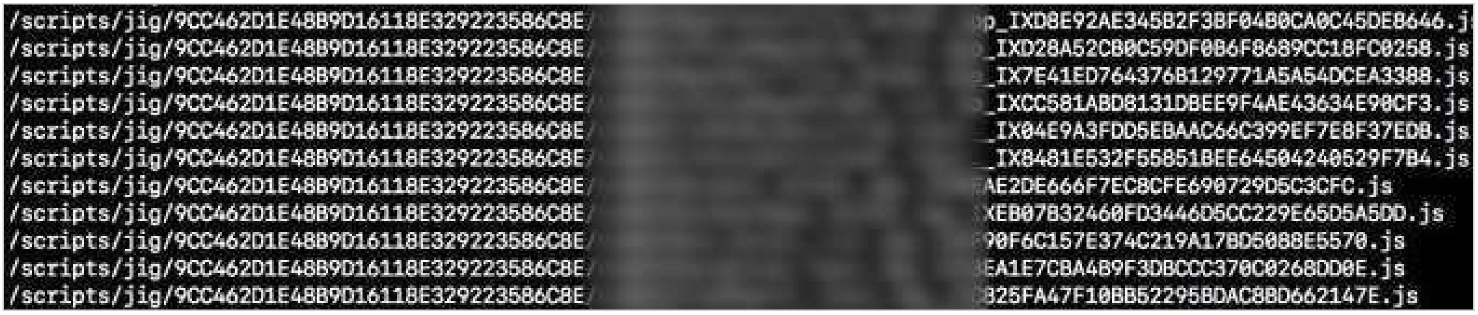
Testing Process

Since there were no root checks we were able to browse through internal storage and find Chromium's cache.



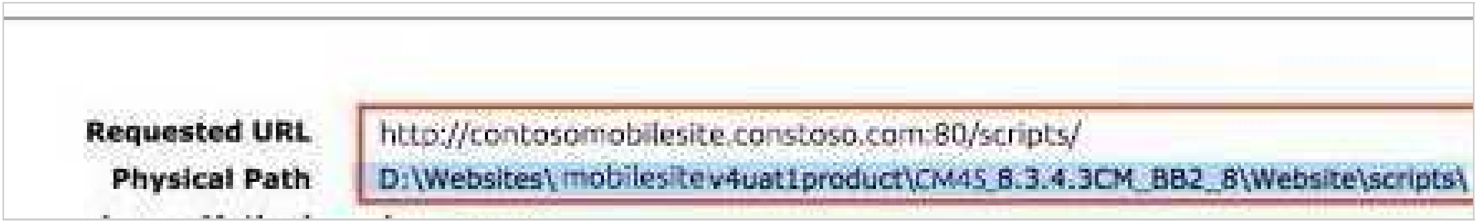
```
root@android:/data/data/com.contoso.ContosoMobile/cache/webviewCacheChromium # ls
data_0
data_1
data_2
data_3
f_000001
f_000002
f_000003
f_000004
f_000005
f_000006
f_000007
f_000008
f_000009
```

Inside the internal storage we were able to identify various scripts in use - which provided us insight into the application.



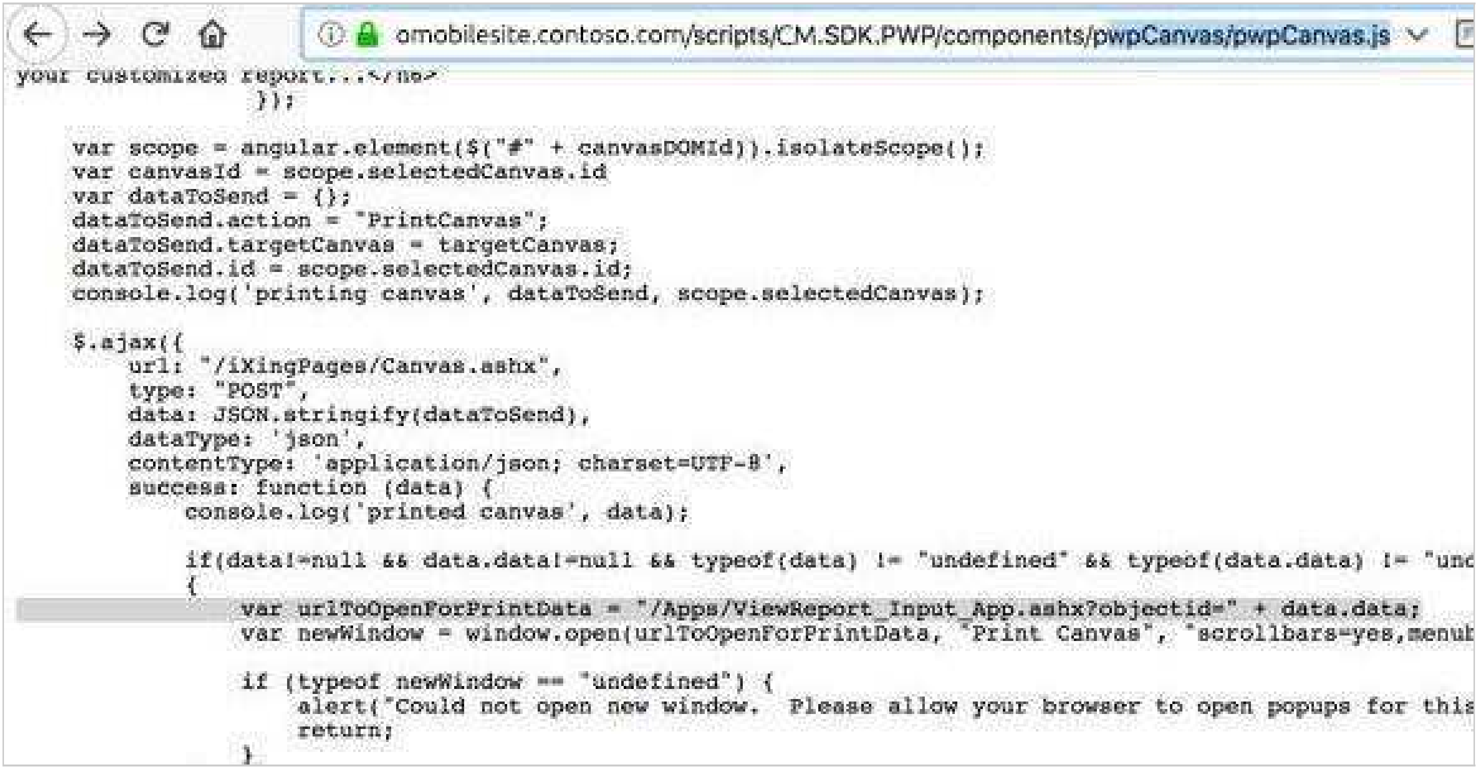
```
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]p_IXD8E92AE345B2F3BF04B0CA0C45DE8646.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]_IXD28A52CB0C59DF0B6F8689CC18FC0258.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]_IX7E41ED764376B129771A5A54DCEA3388.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]_IXCC581ABD8131DBEE9F4AE43634E98CF3.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]_IX04E9A3FDD5EBAAC66C399EF7E8F37EDB.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]_IX8481E532F55851BEE64504240529F7B4.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]AE2DE666F7EC8CFE690729D5C3CFC.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]XEB07B32460FD3446D5CC229E65D5A5DD.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]90F6C157E374C219A17BD5088E5570.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]EA1E7CBA4B9F3DBCCC370C0268DD0E.js
/scripts/jig/9CC462D1E48B9D16118E329223586C8E/[redacted]325FA47F10BB52295BDAC8BD662147E.js
```

Using a web browser, we navigated to the main folder for scripts - which gave us detailed error information.



Requested URL	http://contosomobilesite.constoso.com:80/scripts/
Physical Path	D:\Websites\mobilesitev4uat1product\CM45_8.3.4.3CM_BB2_8\Website\scripts\

Additionally, we navigated to a script found via this method and could see the code in use.



omobilesite.contoso.com/scripts/CM.SDK.PWP/components/pwpCanvas/pwpCanvas.js

```
your customized report...</h5>
                    });

        var scope = angular.element($("#" + canvasDOMId)).isolateScope();
        var canvasId = scope.selectedCanvas.id
        var dataToSend = {};
        dataToSend.action = "PrintCanvas";
        dataToSend.targetCanvas = targetCanvas;
        dataToSend.id = scope.selectedCanvas.id;
        console.log('printing canvas', dataToSend, scope.selectedCanvas);

        $.ajax({
            url: "/iXingPages/Canvas.ashx",
            type: "POST",
            data: JSON.stringify(dataToSend),
            dataType: 'json',
            contentType: 'application/json; charset=UTF-8',
            success: function (data) {
                console.log('printed canvas', data);

                if(data!=null && data.data!=null && typeof(data) != "undefined" && typeof(data.data) != "und
                {
                    var urlToOpenForPrintData = "/Apps/ViewReport_Input_App.ashx?objectid=" + data.data;
                    var newWindow = window.open(urlToOpenForPrintData, "Print Canvas", "scrollbars=yes,menub

                    if (typeof newWindow == "undefined") {
                        alert("Could not open new window.  Please allow your browser to open popups for this
                        return;
                    }
```

H1 No SSL Certificate Pinning

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **High**

Description

Certificate pinning is the process of associating a backend server with a particular x509 certificate or public key instead of accepting any certificate signed by a trusted certificate authority. After storing ("pinning") the server certificate or public key, the mobile app will subsequently connect only to the known server. Withdrawing trust from external certificate authorities reduces the attack surface of the application. Certificate pinning is used to verify that client communications are being sent to the intended remote server, so if the server presents an unknown certificate, the application will not transmit the data. Without pinning, an attacker can implement their own intercepting proxy with an arbitrary certificate to view requests to and from their client device in cleartext.

Affected Application(s) & Platforms

Contoso Mobile Client: Android

Contoso Mobile Client: iOS

Remediation

Implement SSL certificate pinning in all mobile applications.

The Android developer documentation has a good write-up on HTTPS, SSL, and certificate pinning:

- https://developer.android.com/training/articles/security-ssl.html

This blog post gives an in-depth explanation of how to implement pinning in iOS applications:

- https://infinum.co/the-capsized-eight/how-to-make-your-ios-apps-more-secure-with-ssl-pinning

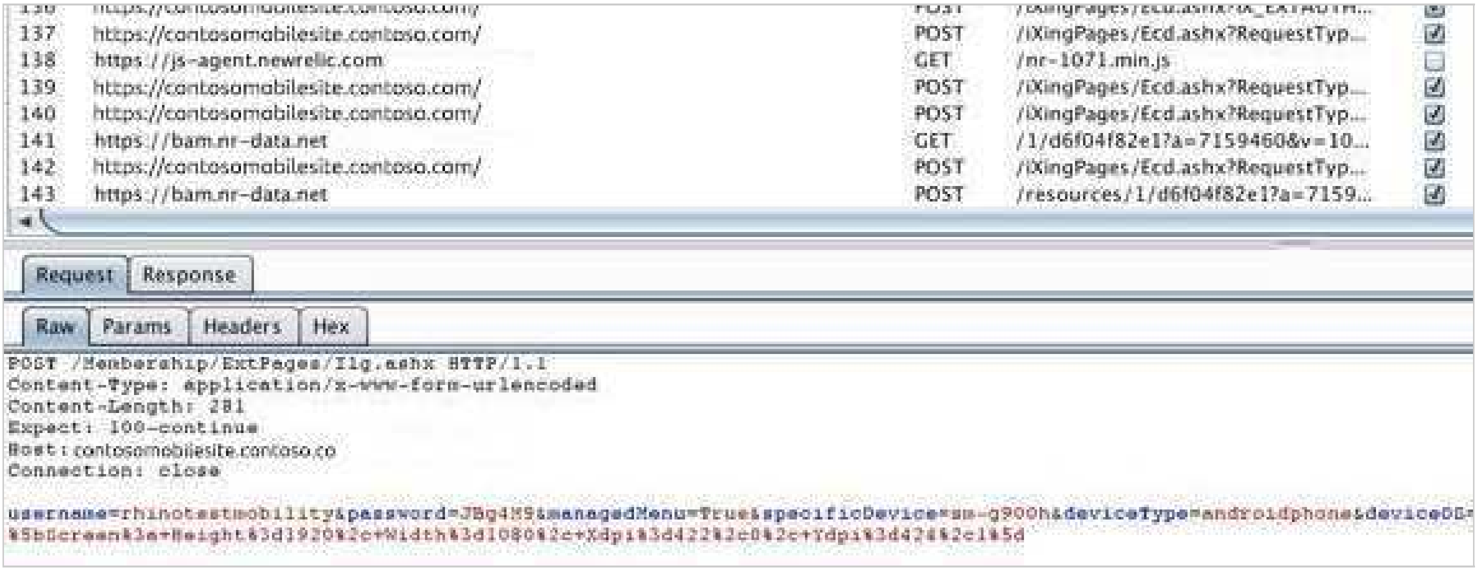
There are also many pre-built libraries to handle certificate pinning such as TrustKit (iOS) and OkHTTP (android). We've provided links to these in the "See More" sections. There are many other libraries out there to assist with pinning as well, including a plugin for the popular Apache Cordova framework.

For situations where the certificate pinning bypass is on the actual operating system level rather than the application, such as when using SSL Kill Switch 2 on a jailbroken iOS device, the best protection would be to implement rooted Android/jailbroken iOS device checks and to not allow the application to run if those checks detect a modified device.

Testing Process

This vulnerability was discovered by decompiling the application and not finding any functionality to implement SSL certificate pinning. This was then confirmed by forcing the application traffic through an interception proxy (Burp Suite) and successfully viewing requests and responses.

Below is an interception proxy being used by the assessor to intercept and modify requests.



```
137  https://contosomobilesite.contoso.com/   POST  /iXingPages/Ecd.ashx?RequestTyp...  ✓
138  https://js-agent.newrelic.com            GET   /nr-1071.min.js
139  https://contosomobilesite.contoso.com/   POST  /iXingPages/Ecd.ashx?RequestTyp...  ✓
140  https://contosomobilesite.contoso.com/   POST  /iXingPages/Ecd.ashx?RequestTyp...  ✓
141  https://bam.nr-data.net                  GET   /1/d6f04f82e1?a=7159460&v=10...     ✓
142  https://contosomobilesite.contoso.com/   POST  /iXingPages/Ecd.ashx?RequestTyp...  ✓
143  https://bam.nr-data.net                  POST  /resources/1/d6f04f82e1?a=7159...   ✓
```

```
POST /Membership/ExtPages/Ilg.ashx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 281
Expect: 100-continue
Host: contosomobilesite.contoso.co
Connection: close

username=rhinotestmobility&password=JBg4M9&managedMenu=True&specificDevice=sm-g900h&deviceType=androidphone&deviceOS=
%5bScreen%3a+Height%3d1920%2c+Width%3d1080%2c+Xdpi%3d422%2c0%2c+Ydpi%3d424%2c1%5d
```

H2 Username Enumeration via Password Reset

Risk Rating: High

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description
This vulnerability is discovered by submitting both known valid and invalid usernames/emails to the password reset functionality of the application, and comparing application responses. If the invalid user is identified through the app error message, an attacker is able to enumerate a list of emails used by brute-forcing possible emails and noting the differences in responses.

Affected Application(s) & Platforms

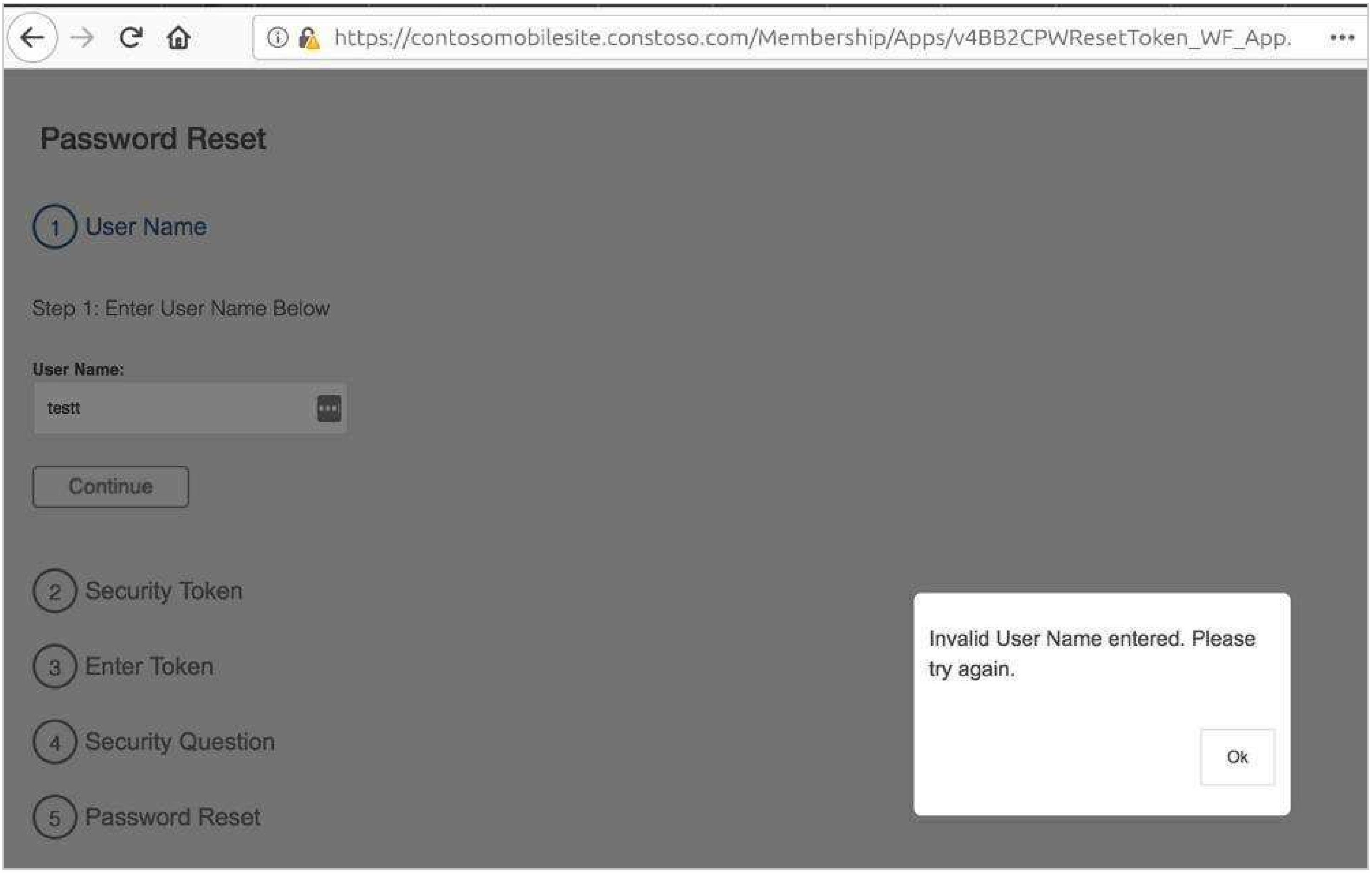
Contoso Mobile Client: Android

Contoso Mobile Client: iOS

Remediation
Change password recovery messages to be generic, so as to not indicate if an email was sent (username exists in system) or there was an error (username does not exist in the system).

Testing Process
This vulnerability was confirmed by providing an incorrect username and requesting for password reset. HTTP Response 500 can be reliably used to enumerate usernames and which Email IDs have accounts on the application.

Upon entering an invalid username into the password reset form the following message is displayed.



https://contosomobilesite.constoso.com/Membership/Apps/v4BB2CPWResetToken_WF_App.

Password Reset

1 User Name

Step 1: Enter User Name Below

User Name:

testt

Continue

2 Security Token

3 Enter Token

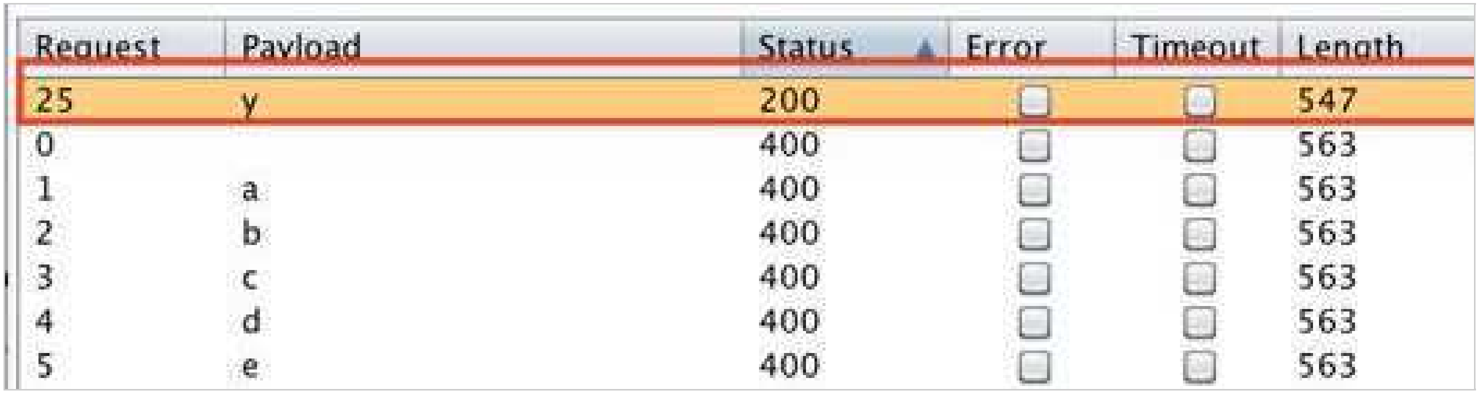
4 Security Question

5 Password Reset

Invalid User Name entered. Please try again.

Ok

Through an intercepting proxy we can see that incorrect guesses for a username result in a status 400 and the correct username results in a status 200.



Request	Payload	Status	Error	Timeout	Length
25	y	200	☐	☐	547
0		400	☐	☐	563
1	a	400	☐	☐	563
2	b	400	☐	☐	563
3	c	400	☐	☐	563
4	d	400	☐	☐	563
5	e	400	☐	☐	563

H3 iOS App Transport Security (ATS) Disabled

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **High**

Description

App Transport Security (ATS) is an iOS feature that forces mobile apps to connect to back-end servers using HTTPS, instead of HTTP, to encrypt data in transit. ATS enforces a minimum security level for communications between a mobile app and web services that support its functionality. Software development kits (SDKs) for iOS 9.0 or later enable ATS by default, but developers can opt-out by setting an app's NSAllowsArbitraryLoads key to true.

Affected Application(s) & Platforms

Contoso Mobile Client: iOS

Remediation

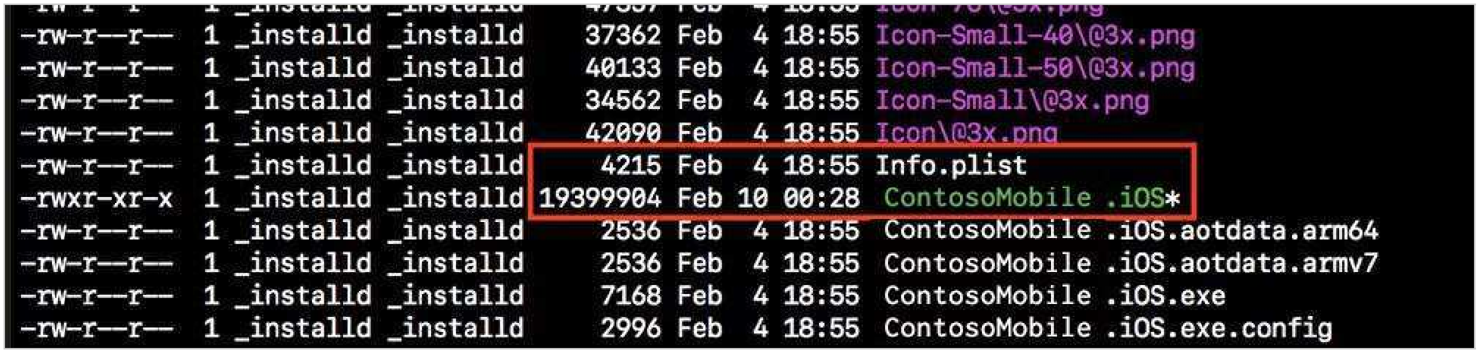
Ensure that the NSAllowsArbitraryLoads setting is set to false in the applications Info.plist file. The application is then forced to connect to the web via HTTPS with the strongest TLS version and cipher suites.

It is recommended to use HTTPS exclusively for all mobile application API calls. In addition, the communication through higher-level APIs needs to be encrypted using TLS version 1.2 with forward secrecy. If the application tries to make a connection that doesn't follow this requirement, an error is thrown. If the application needs to make a request to an insecure domain, it must be specified in the applications "Info.plist" file.

Testing Process

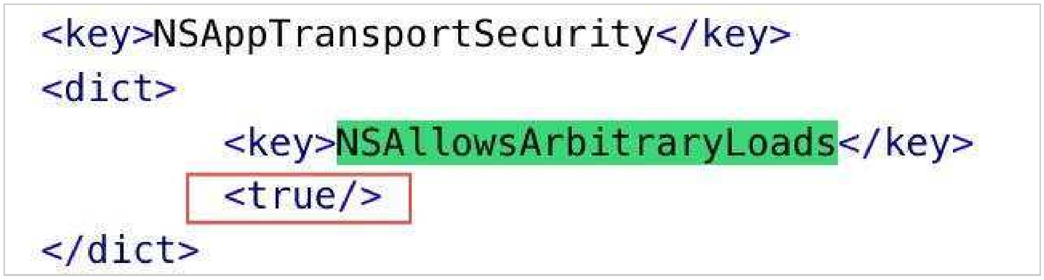
The assessor reviewed the security-related settings in the applications source code and discovered this vulnerability.

The assessor located the "Info.plist" file



```
-rw-r--r--  1 _installd _installd    37362 Feb  4 18:55 Icon-Small-40\@3x.png
-rw-r--r--  1 _installd _installd    40133 Feb  4 18:55 Icon-Small-50\@3x.png
-rw-r--r--  1 _installd _installd    34562 Feb  4 18:55 Icon-Small\@3x.png
-rw-r--r--  1 _installd _installd    42090 Feb  4 18:55 Icon\@3x.png
-rw-r--r--  1 _installd _installd     4215 Feb  4 18:55 Info.plist
-rwxr-xr-x  1 _installd _installd 19399904 Feb 10 00:28  ContosoMobile .iOS*
-rw-r--r--  1 _installd _installd     2536 Feb  4 18:55  ContosoMobile .iOS.aotdata.arm64
-rw-r--r--  1 _installd _installd     2536 Feb  4 18:55  ContosoMobile .iOS.aotdata.armv7
-rw-r--r--  1 _installd _installd     7168 Feb  4 18:55  ContosoMobile .iOS.exe
-rw-r--r--  1 _installd _installd     2996 Feb  4 18:55  ContosoMobile .iOS.exe.config
```

Upon manual inspection the "NSAllowsArbitraryLoads" key was set to True.



```
<key>NSAppTransportSecurity</key>
<dict>
        <key>NSAllowsArbitraryLoads</key>
        <true/>
</dict>
```

M1 Sensitive Information Disclosure via SQLite Database

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

The mobile application uses an unencrypted SQLite database.This database can be accessed by an attacker with physical access to the mobile device or a malicious application with root access to the device. The application should not store sensitive information in clear text. An attacker can take advantage of the lack of this validation, to perform additional tests in search of vulnerabilities and to gain more information about how the application works.

Affected Application(s) & Platforms

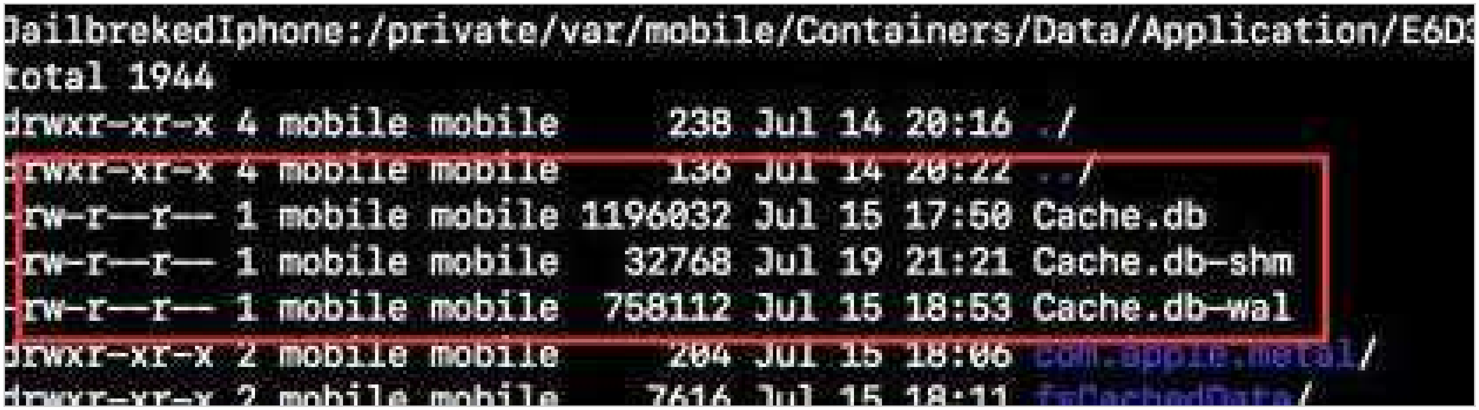
Contoso Mobile Client: iOS

Remediation

We recommend avoiding the storage of internal information, unless it is strictly necessary. In this case, we recommend to encrypt the information inside the SQLite database. Additionally, we recommend to avoid storing the encryption key inside the internal storage.

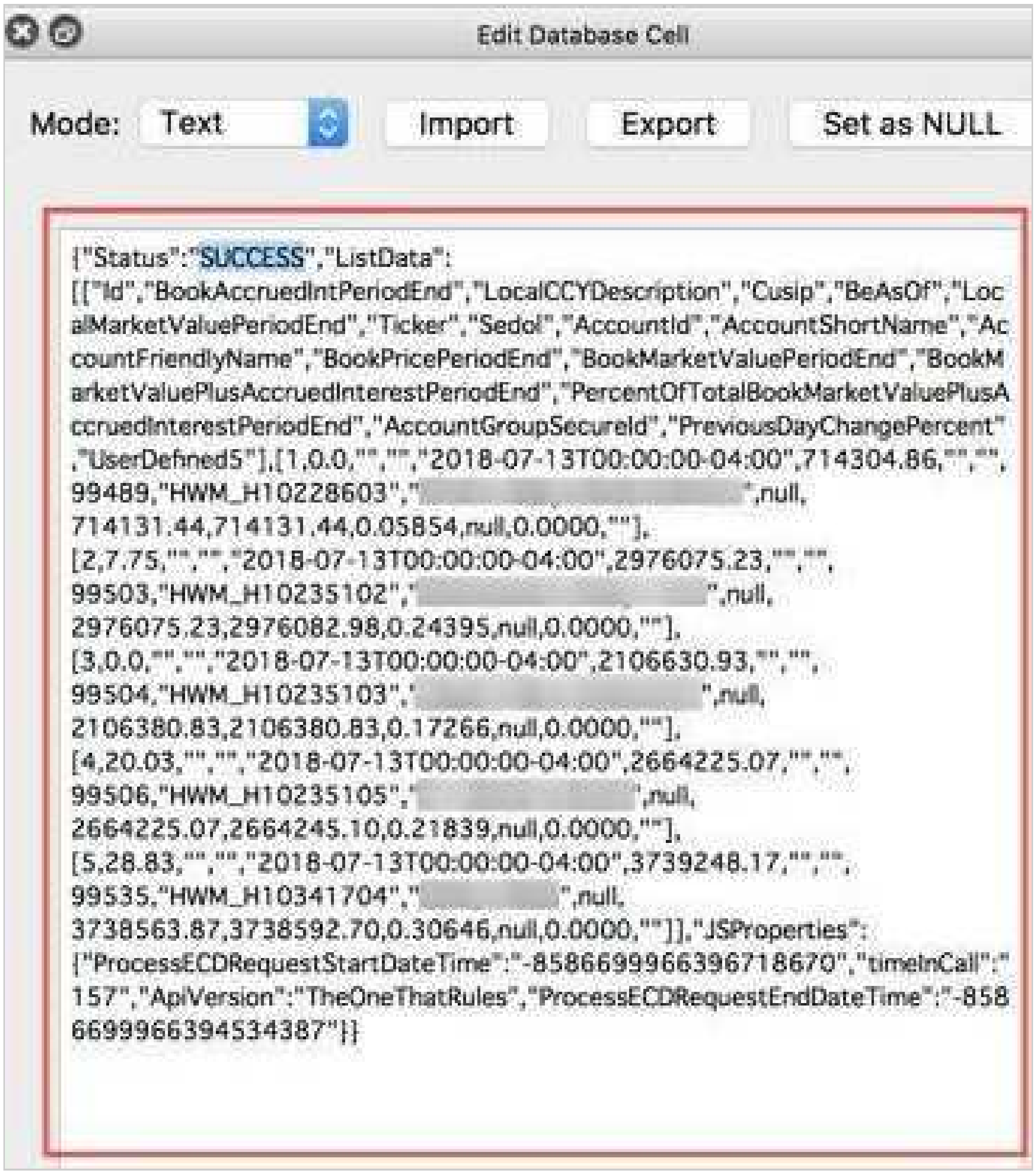
Testing Process

The assessor tested this by exploring the application files and finding a database that was unencrypted.



```
JailbrekedIphone:/private/var/mobile/Containers/Data/Application/E6D3
total 1944
drwxr-xr-x 4 mobile mobile     238 Jul 14 20:16 ./
drwxr-xr-x 4 mobile mobile     136 Jul 14 20:22 ../
-rw-r--r-- 1 mobile mobile 1196032 Jul 15 17:50 Cache.db
-rw-r--r-- 1 mobile mobile   32768 Jul 19 21:21 Cache.db-shm
-rw-r--r-- 1 mobile mobile  758112 Jul 15 18:53 Cache.db-wal
drwxr-xr-x 2 mobile mobile     204 Jul 15 18:06 com.apple.metal/
drwxr-xr-x 2 mobile mobile    7616 Jul 15 18:11 fsCachedData/
```

Browsing through the database there was information that does not need to be made accessible to the user.



Mode: Text

```
{"Status":"SUCCESS","ListData":
[["Id","BookAccruedIntPeriodEnd","LocalCCYDescription","Cusip","BeAsOf","Loc
alMarketValuePeriodEnd","Ticker","Sedol","AccountId","AccountShortName","Ac
countFriendlyName","BookPricePeriodEnd","BookMarketValuePeriodEnd","BookM
arketValuePlusAccruedInterestPeriodEnd","PercentOfTotalBookMarketValuePlusA
ccruedInterestPeriodEnd","AccountGroupSecureId","PreviousDayChangePercent"
,"UserDefined5"],[1,0.0,"","","2018-07-13T00:00:00-04:00",714304.86,"","",
99489,"HWM_H10228603","[redacted]",null,
714131.44,714131.44,0.05854,null,0.0000,""],
[2,7.75,"","","2018-07-13T00:00:00-04:00",2976075.23,"","",
99503,"HWM_H10235102","[redacted]",null,
2976075.23,2976082.98,0.24395,null,0.0000,""],
[3,0.0,"","","2018-07-13T00:00:00-04:00",2106630.93,"","",
99504,"HWM_H10235103","[redacted]",null,
2106380.83,2106380.83,0.17266,null,0.0000,""],
[4,20.03,"","","2018-07-13T00:00:00-04:00",2664225.07,"","",
99506,"HWM_H10235105","[redacted]",null,
2664225.07,2664245.10,0.21839,null,0.0000,""],
[5,28.83,"","","2018-07-13T00:00:00-04:00",3739248.17,"","",
99535,"HWM_H10341704","[redacted]",null,
3738563.87,3738592.70,0.30646,null,0.0000,""]],"JSProperties":
{"ProcessECDRequestStartDateTime":"-8586699966396718670","timeInCall":"
157","ApiVersion":"TheOneThatRules","ProcessECDRequestEndDateTime":"-858
6699966394534387"}}
```

M2 No Root Check on Android Application

Risk Rating: **Medium**

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

No root detection is implemented in the application. This allows an attacker to modify the app on a rooted Android device, which means that the attacker can potentially induce behaviors that otherwise would not occur. Examples include removing ASLR, overwriting critical functions, and an overall wider attack surface.

For more information, visit the following links:

- https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#testing-root-detection
- https://developer.android.com/training/safetynet/attestation.html
- https://developers.google.com/android/reference/com/google/android/gms/safetynet/SafetyNet

Affected Application(s) & Platforms

Contoso Mobile Client: Android

Remediation

Implement root detection before beginning the runtime of the application.

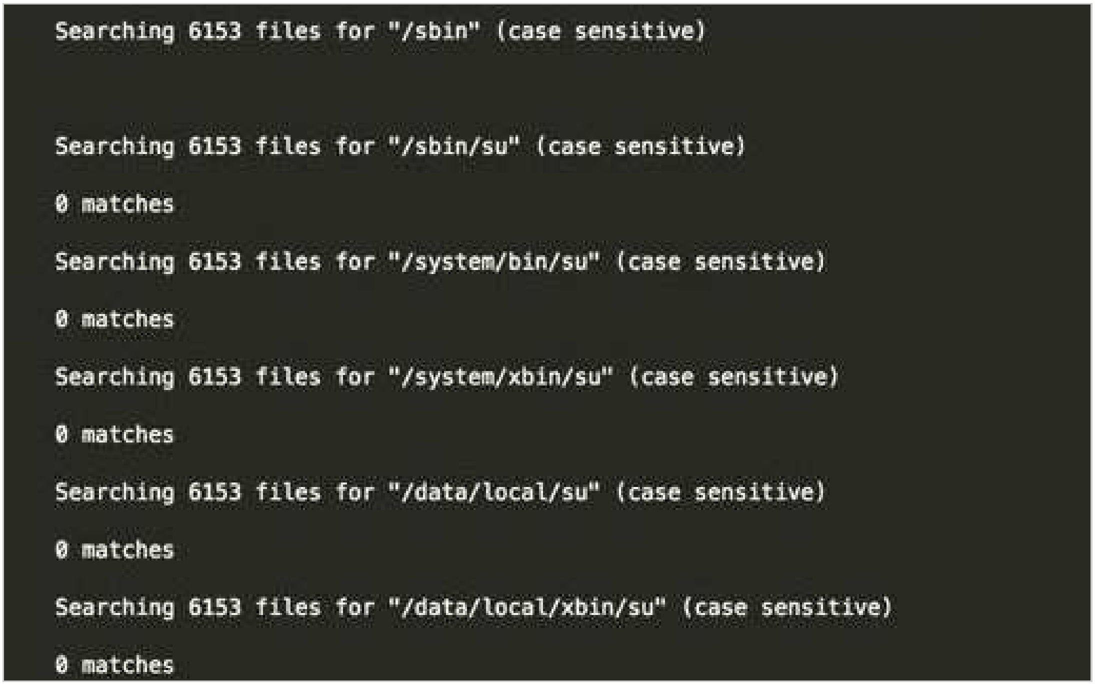
There are many ways to detect a rooted Android device. The main idea is to try and perform an action that would only be allowed on a rooted device, then if it was successful, the app knows that it is running on a rooted device. There are often bypasses for certain rooted device checks, so it is recommended to implement as many root checks as possible to increase the difficulty of a rooted device bypassing the checks. Some examples of those different checks are listed on the next page:

- Check for known root applications, such as:
  - com.noshufou.android.su
  - com.thirdparty.superuser
  - eu.chainfire.supersu
  - com.koushikdutta.superuser
- Check for existing `su` binaries, such as:
  - /system/bin/su
  - /system/xbin/su
  - /sbin/su
  - /system/su
  - /system/bin/.ext/.su
- Check for OTA certificates at `/etc/security/otacerts.zip`.
- Attempt to use the `su` command directly and compare the current user ID before and after to see if the user was successfully upgraded to root.
- Utilize SafetyNet by Google. This was developed to try and detect device modifications and will prevent the application from running if the device fails the checks.
- Apache Cordova also has plugins to run these kinds of checks.

Testing Process

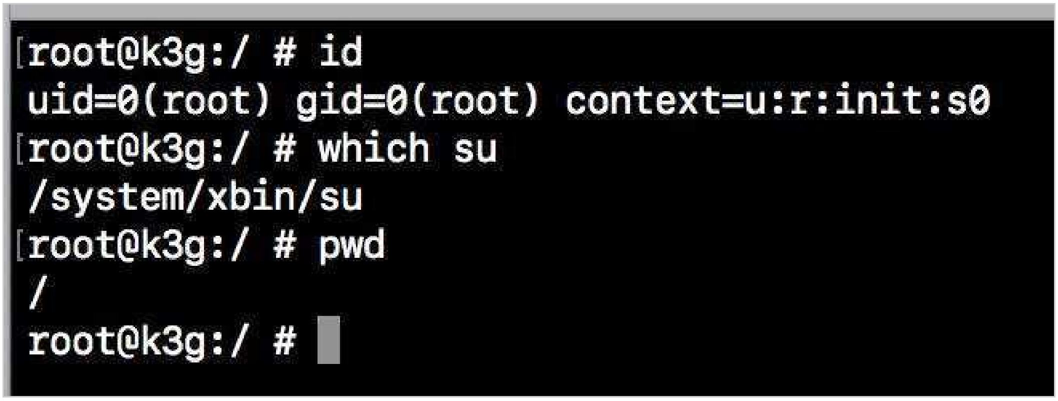
This was discovered by running the application on both a rooted Android device and on a non-rooted Android device and not observing any differences in behavior between the two.

We first began by searching for common rootchecks and noting there were none.



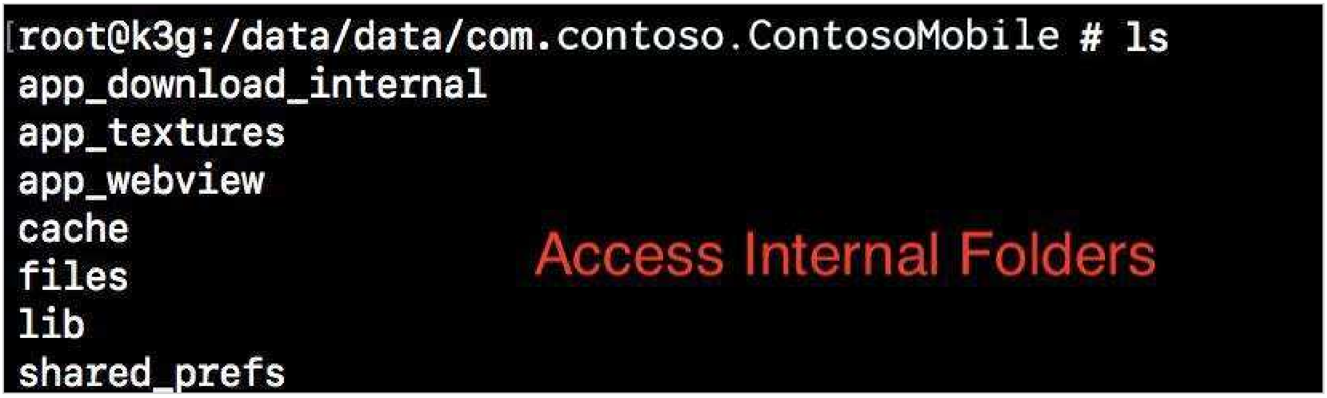
```
Searching 6153 files for "/sbin" (case sensitive)


Searching 6153 files for "/sbin/su" (case sensitive)

0 matches

Searching 6153 files for "/system/bin/su" (case sensitive)

0 matches

Searching 6153 files for "/system/xbin/su" (case sensitive)

0 matches

Searching 6153 files for "/data/local/su" (case sensitive)

0 matches

Searching 6153 files for "/data/local/xbin/su" (case sensitive)

0 matches
```

Afterwards, we switched users to root.



```
[root@k3g:/ # id
uid=0(root) gid=0(root) context=u:r:init:s0
[root@k3g:/ # which su
/system/xbin/su
[root@k3g:/ # pwd
/
root@k3g:/ #
```

Then we attempted to access internal folders (which was permitted).



```
root@k3g:/data/data/com.contoso.ContosoMobile # ls
app_download_internal
app_textures
app_webview
cache
files
lib
shared_prefs
```

M3 Unprotected Android Service

CWE-926

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description

If a Service is exported and not protected with strong permissions, then any application can start and bind to the Service. Depending on the duties of a particular Service, it may leak information or perform unauthorized tasks.

Services can be exported both explicitly and implicitly. By default, Services are not exported, but if the particular Service has an intent filter defined in it, then it is automatically (implicitly) exported. They can also be exported by directly setting the `android:exported` property to true for the particular Service, in the "AndroidManifest.xml" file.

The Services that are being exported, either explicitly or implicitly, are listed here:

- place
- holder
- here

Affected Application(s) & Platforms

Contoso Mobile Client: Android

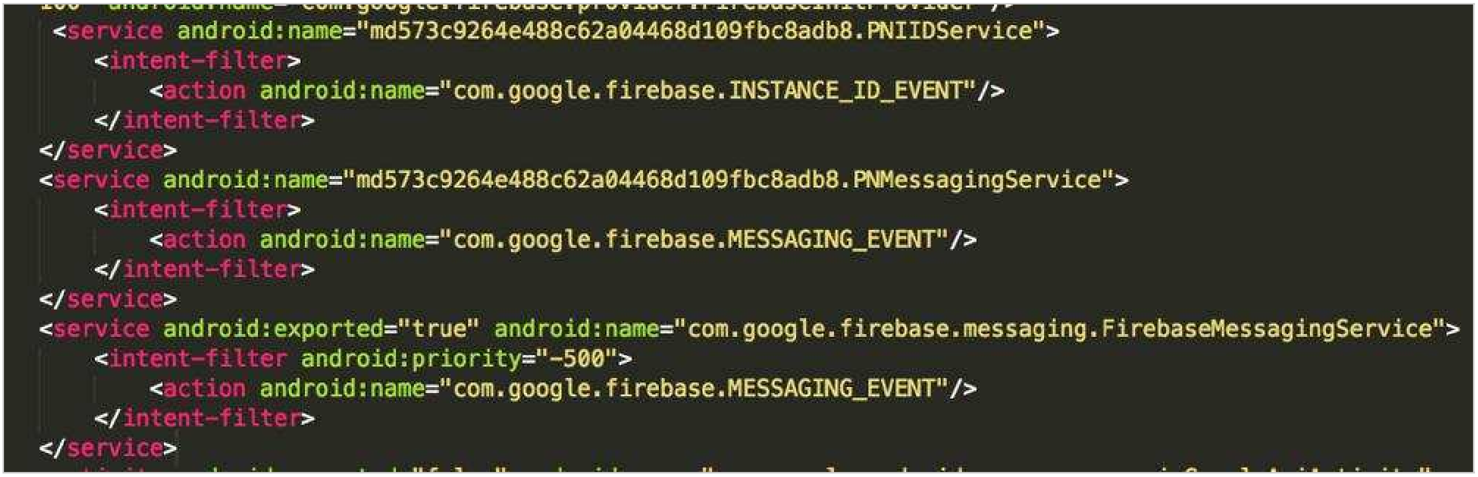
Remediation

Explicitly set the `android:exported` property to false in the AndroidManifest.xml file. However, the `android:exported` attribute is not the only way to limit a Services exposure to other applications. The application can also impose permission-based restrictions by defining custom permissions for a Service. This is helpful if the developer wants to limit the access to the applications components to those apps which have permission.

Testing Process

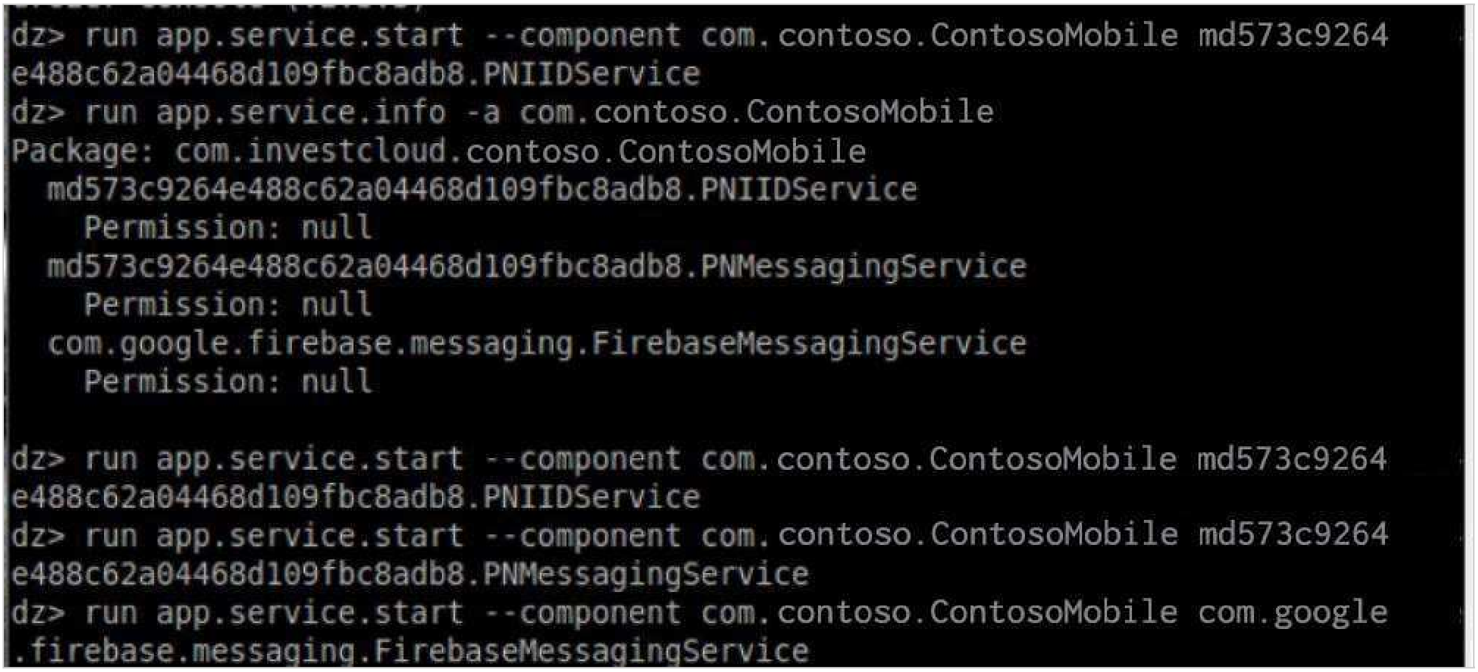
This vulnerability was detected by decompiling the application and going through the "AndroidManifest.xml" file. This was confirmed by using the application drozer to try and invoke services that were exported by the affected application from itself.

The following image has the services exported and none have an assigned application permission.



```
 <service android:name="md573c9264e488c62a04468d109fbc8adb8.PNIIDService">
     <intent-filter>
         <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
     </intent-filter>
</service>
<service android:name="md573c9264e488c62a04468d109fbc8adb8.PNMessagingService">
     <intent-filter>
         <action android:name="com.google.firebase.MESSAGING_EVENT"/>
     </intent-filter>
</service>
<service android:exported="true" android:name="com.google.firebase.messaging.FirebaseMessagingService">
     <intent-filter android:priority="-500">
         <action android:name="com.google.firebase.MESSAGING_EVENT"/>
     </intent-filter>
</service>
```

Invoking the services that were exported by the application.



```
dz> run app.service.start --component com.contoso.ContosoMobile md573c9264
e488c62a04468d109fbc8adb8.PNIIDService
dz> run app.service.info -a com.contoso.ContosoMobile
Package: com.investcloud.contoso.ContosoMobile
  md573c9264e488c62a04468d109fbc8adb8.PNIIDService
    Permission: null
  md573c9264e488c62a04468d109fbc8adb8.PNMessagingService
    Permission: null
  com.google.firebase.messaging.FirebaseMessagingService
    Permission: null

dz> run app.service.start --component com.contoso.ContosoMobile md573c9264
e488c62a04468d109fbc8adb8.PNIIDService
dz> run app.service.start --component com.contoso.ContosoMobile md573c9264
e488c62a04468d109fbc8adb8.PNMessagingService
dz> run app.service.start --component com.contoso.ContosoMobile com.google
.firebase.messaging.FirebaseMessagingService
```

M4 No Jailbreak Check On iOS Application

Risk Rating: Medium

Exploitation Likelihood: **High** | Potential Impact: **Medium**

Description

No jailbreak detection is implemented in the application. This allows an attacker to modify the app on a jailbroken iOS device, which means the attacker can potentially induce behaviors that otherwise would not occur. Examples of the impact include removing ASLR, overwriting critical functions, and an overall wider attack surface.

Affected Application(s) & Platforms

Contoso Mobile Client: iOS

Remediation

Implement jailbreak detection before beginning the runtime of your application. Often times there are bypasses that users can use to avoid having their jailbroken device detected, so it is recommended to implement as many different checks as possible. Some different methods to check for a jailbroken device are listed below:

- Check for the existence of common files or directories that are associated with a jailbroken device. A list of some of the potential files to check can be seen here:
  - /Applications/Cydia.app
  - /Applications/FakeCarrier.app
  - /Applications/Icy.app
  - /Applications/IntelliScreen.app
  - /Applications/MxTube.app
  - /Applications/RockApp.app
  - /Applications/SBSettings.app
  - /Applications/WinterBoard.app
  - /Applications/blackra1n.app
  - /Library/MobileSubstrate/DynamicLibraries/LiveClock.plist
  - /Library/MobileSubstrate/DynamicLibraries/Veency.plist

  - /Library/MobileSubstrate/MobileSubstrate.dylib
  - /System/Library/LaunchDaemons/com.ikey.bbot.plist
  - /System/Library/LaunchDaemons/com.saurik.Cydia.Startup.plist
  - /bin/bash
  - /bin/sh
  - /etc/apt
  - /etc/ssh/sshd_config
  - /private/var/lib/apt
  - /private/var/lib/cydia
  - /private/var/mobile/Library/SBSettings/Themes
  - /private/var/stash
  - /private/var/tmp/cydia.log
  - /usr/bin/sshd
  - /usr/libexec/sftp-server
  - /usr/libexec/ssh-keysign
  - /usr/sbin/sshd
  - /var/cache/apt
  - /var/lib/apt
  - /var/lib/cydia
- Attempt to write to a file in a location that is outside of the applications virtual sandbox. An example would be to try and write to the /private directory. This attempt would fail on a non-jailbroken device because the application does not have permission to access anything outside of its virtual sandbox, but if the application succeeds in writing to that location, then it can be deduced that it has root permissions, which means that the device is jailbroken. Example code for this kind of check can be seen here:

```
NSError *error;
NSString *stringToBeWritten = @"This is a test.";
[stringToBeWritten writeToFile:@"/private/jailbreak.txt" atomically:YES
   encoding:NSUTF8StringEncoding error:&error];
if(error==nil){
   //Device is jailbroken
   return YES;
} else {
   //Device is not jailbroken
   [[NSFileManager defaultManager] removeItemAtPath:@"/private/jailbreak.txt" error:n
il];
}
```

- Check the available protocol handlers. Nearly every jailbroken iOS device in existence includes the Cydia app store.

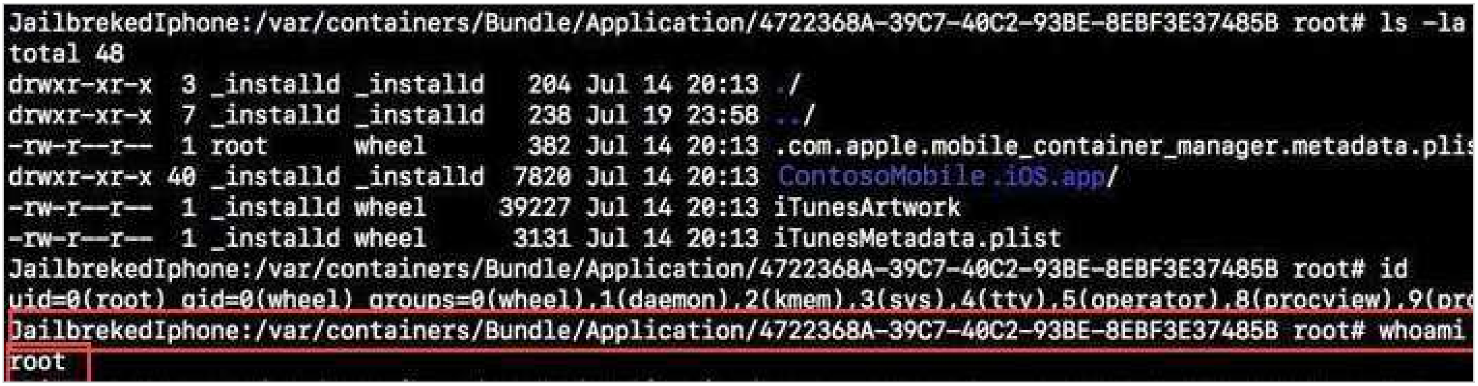
When installed, Cydia will install the `cydia://` protocol handler. If a mobile application is able to successfully use the Cydia protocol handler, then it means that the device is jailbroken.

- Call the system() function with a NULL argument and analyze the response, which will be 0 if the device is not jailbroken or 1 if the device is jailbroken. This happens because the system() function checks whether `/bin/sh` can be accessed and it only can be on jailbroken devices.
- There are also some plugins that can be installed into mobile applications to take over the responsibility of these checks.

Testing Process

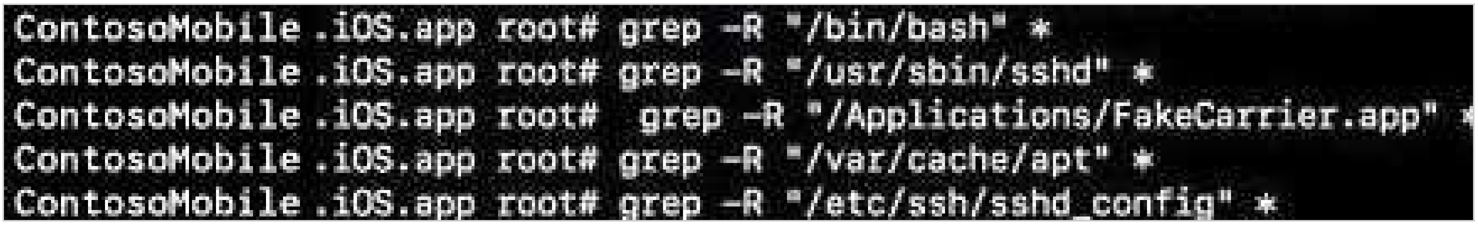
This was discovered by running the application both on a jailbroken iOS device and a non-jailbroken iOS device and not noticing any differences in the behavior of the application.

In the following screenshot the assessor is running the application on a jailbroken iOS device.



```
JailbrekedIphone:/var/containers/Bundle/Application/4722368A-39C7-40C2-93BE-8EBF3E37485B root# ls -la
total 48
drwxr-xr-x  3 _installd _installd   204 Jul 14 20:13 ./
drwxr-xr-x  7 _installd _installd   238 Jul 19 23:58 ../
-rw-r--r--  1 root      wheel       382 Jul 14 20:13 .com.apple.mobile_container_manager.metadata.plis
drwxr-xr-x 40 _installd _installd  7820 Jul 14 20:13 ContosoMobile.iOS.app/
-rw-r--r--  1 _installd wheel     39227 Jul 14 20:13 iTunesArtwork
-rw-r--r--  1 _installd wheel      3131 Jul 14 20:13 iTunesMetadata.plist
JailbrekedIphone:/var/containers/Bundle/Application/4722368A-39C7-40C2-93BE-8EBF3E37485B root# id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(pro
JailbrekedIphone:/var/containers/Bundle/Application/4722368A-39C7-40C2-93BE-8EBF3E37485B root# whoami
root
```

The assessor then searched for common checks against rooted devices which returned nothing.



```
ContosoMobile.iOS.app root# grep -R "/bin/bash" *
ContosoMobile.iOS.app root# grep -R "/usr/sbin/sshd" *
ContosoMobile.iOS.app root#  grep -R "/Applications/FakeCarrier.app" *
ContosoMobile.iOS.app root# grep -R "/var/cache/apt" *
ContosoMobile.iOS.app root# grep -R "/etc/ssh/sshd_config" *
```

L1 Application with "allowBackup" Flag Set to True

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Low**

Description

Android provides an attribute called allowBackup to back up all your application data. This attribute is set in the AndroidManifest.xml file. If the value of this attribute is true, the device allows users to back up the application with Android Debug Bridge (ADB) via the command `$ adb backup` . It allows users who have enabled USB debugging to copy application data off of the device.

To prevent the app data backup, set the android:allowBackup attribute to false ([android:allowBackup=false]). When this attribute is unavailable, the allowBackup setting is enabled by default, and backup must be manually deactivated.

Affected Application(s) & Platforms

Contoso Mobile Client: Android

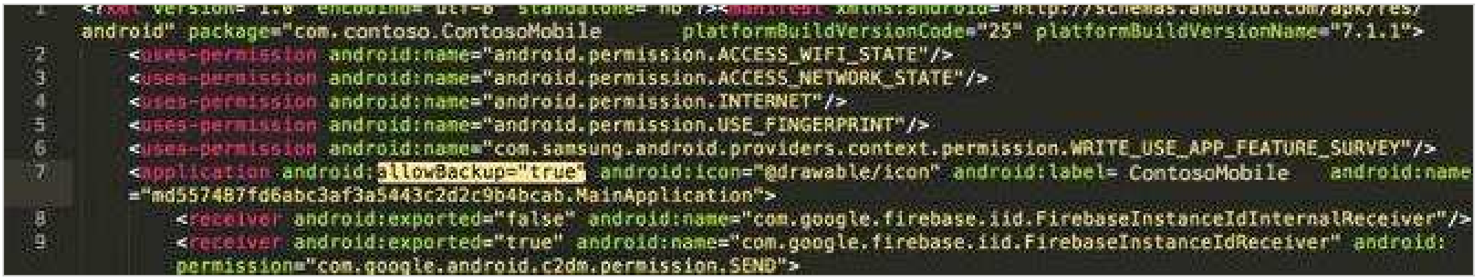
Remediation

Inside the "AndroidManifest.xml" modify the flag "android:allowBackup" from "True" to "False".

Testing Process

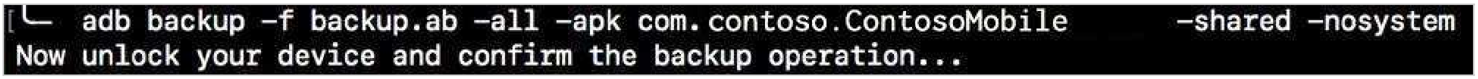
This vulnerability was found by examining the "AndroidManifest.xml" file manually and noting the "allowBackup" attribute was set to "true".

Below is a screenshot of the "AndroidManifest.xml" file.



```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/
android" package="com.contoso.ContosoMobile       platformBuildVersionCode="25" platformBuildVersionName="7.1.1">
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.USE_FINGERPRINT"/>
    <uses-permission android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
    <application android:allowBackup="true" android:icon="@drawable/icon" android:label= ContosoMobile    android:name
    ="md557487fd6abc3af3a5443c2d2c9b4bcab.MainApplication">
        <receiver android:exported="false" android:name="com.google.firebase.iid.FirebaseInstanceIdInternalReceiver"/>
        <receiver android:exported="true" android:name="com.google.firebase.iid.FirebaseInstanceIdReceiver" android:
        permission="com.google.android.c2dm.permission.SEND">
```

After confirming the flag was set to True, the assessor confirmed it was possible using adb to backup the application data.



```
[└  adb backup -f backup.ab -all -apk com.contoso.ContosoMobile       -shared -nosystem
Now unlock your device and confirm the backup operation...
```

L2 Android Application Debugging Enabled

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Low**

Description

The application debugging option is enabled. This allows attackers to hook a debugger to the running application for further, in-depth analysis of how it works. An attacker can dump stack traces and access debugging helper classes, which could potentially lead to bypassing certain application checks or actions to exploit a vulnerability.

Affected Application(s) & Platforms

Contoso Mobile Client: Android

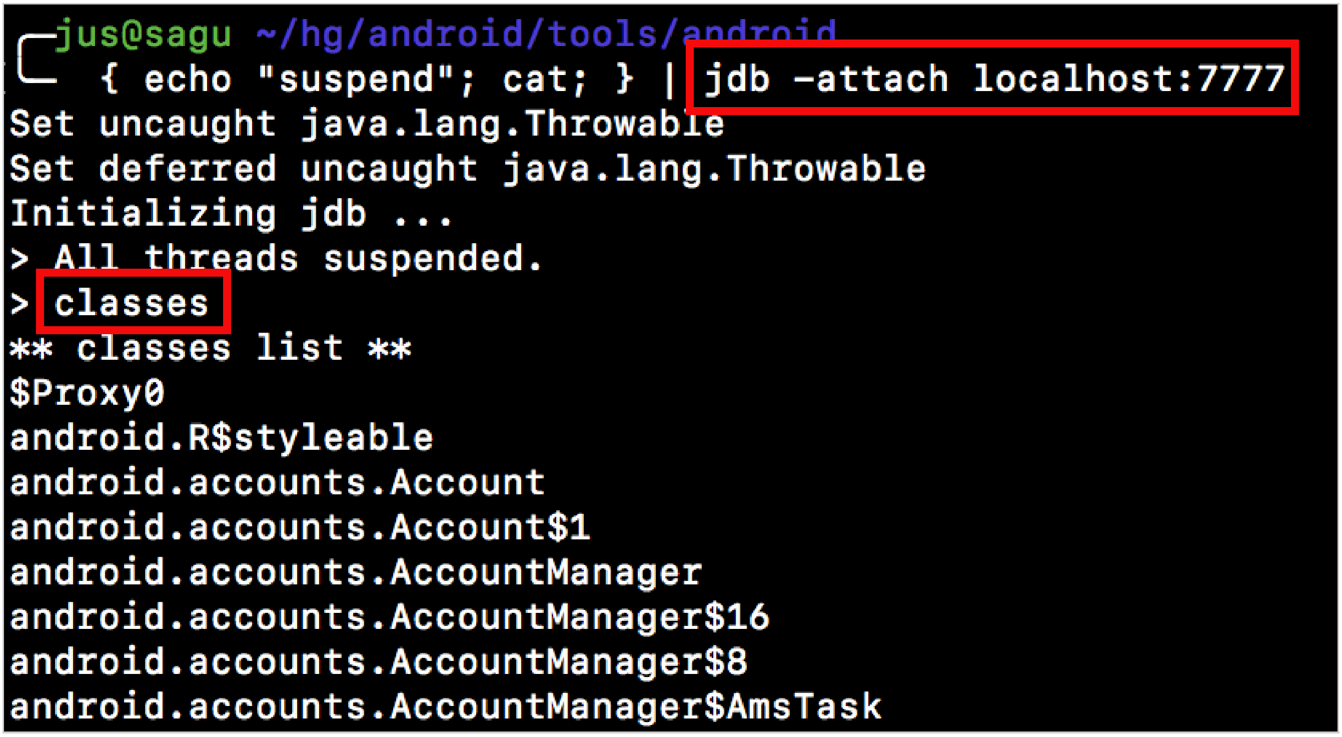
Remediation

Disable application debugging in the "AndroidManifest.xml" file. This can be done by setting the `android:debuggable` property to `false`.

Testing Process

This vulnerability was discovered by decompiling the application and noting the `android:debuggable` property was equal to `true`.

This screenshot shows the assessor hooking Java Debugger (jdb) to the mobile application and then listing the classes that it is using.



```
┌─jus@sagu ~/hg/android/tools/android
└─  { echo "suspend"; cat; } | jdb -attach localhost:7777
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...
> All threads suspended.
> classes
** classes list **
$Proxy0
android.R$styleable
android.accounts.Account
android.accounts.Account$1
android.accounts.AccountManager
android.accounts.AccountManager$16
android.accounts.AccountManager$8
android.accounts.AccountManager$AmsTask
```

RHINO SECURITY LABS TOOLKIT

The software and tools used for security analysis are constantly evolving and changing. To stay at the forefront of industry trends, Rhino Security Labs regularly updates and integrates new tools into its Web Application assessment methodology. Below is the toolset our consultants use during a Web Application assessment.

Burp Suite Professional

Burp Suite is security platform created specifically for the purposes of intensive web application testing. Its capabilities cover the entire vulnerability assessment process, from mapping and analysis of an application to the exploitation of identified vulnerabilities.

Acunetix

An in-depth web application scanner that specializes in doing exhaustive crawling of web-applications as well as detection of a large multitude of common and obscure bugs such as the OWASP Top 10 and many more. It is technology agnostic and can detect bugs in complex technologies such as SOAP/WSDL, SOAP/WCF, REST/WADL, XML, JSON, Google Web Toolkit (GWT) and CRUD operations.

W3af

W3af is an extremely powerful, and flexible framework for finding and exploiting web application vulnerabilities. It is easy to use and extend and features dozens of web assessment and exploitation plugins, which are extensively used by the Rhino Security Labs Team.

Nessus

Nessus is a proprietary vulnerability scanner that specializes in delivering comprehensive mappings of target system vulnerabilities, including web and network vulnerabilities, misconfigurations, weak passwords and even compliance problems, such as with HIPAA and PCI.

Nmap

Nmap is a powerful network security scanning application that uses carefully crafted packets to probe target networks and discover exposed open ports, services, and other host details, such as operating system type.

Nikto

Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6400 potentially dangerous files/CGIs, checks for outdated versions of over 1200 servers, and version specific problems on over 270 servers.

Dirb

DIRB is a Web Content Scanner that looks for existing (and/or hidden) web objects. It functions by launching a dictionary-based attack against a web server and analyzing the response. DIRB searches for specific web objects that other generic CGI scanners often miss, but does not perform vulnerability scans.

Hashcat

Hashcat is the considered world's fastest password recovery tool. It harnesses the power of GPUs and CPUs to bruteforce and crack hashes extracted from a large number of different devices, servers or services.

APPENDIX A: CHANGES TO ENVIRONMENT

The following changes were made to the environment in scope. These do not necessarily represent a significant impact to the environment, but are included for the full accounting of modifications by the penetration testing team at Rhino Security Labs.

NO CHANGES

No changes were made to the environment in scope, such as creating new user accounts or uploading files to the target system. This is provided as the full accounting of modifications by the penetration testing team at Rhino Security Labs.

888.944.8679
info@rhinosecuritylabs.com
464 12th Ave, Suite 300 | Seattle, WA 98122