31 JULY 2017

MOBILE APPLICATION PENETRATION TEST REPORT

Version 1.4

IniTech, Inc.
Bill Lumbergh | Chief Technology Officer

RHINO SECURITY LABS

MOBILE APPLICATION ASSESSMENT INFORMATION

Security Consultant

Dwight Hohnstein
Dwight.hohnstein@rhinosecuritylabs.com
(888) 944-8679

Assessment Prepared for

Bill Lumbergh
Chief Technology Officer
IniTech, Ltd.

Security Engagement Manager

Christopher Lakin
Chris.lakin@rhinosecuritylabs.com
(888) 944-8679

Project Number

05-17-ITL-SE

Assessment Scope Summary

Engagement Timeframe

05/02/2017 – 05/21/2017

Engagement Scope

IniTech Corporate Website
IniTech Administrative Portal
IniTech Customer Portal

Revision History

05-21-2017	Dwight Hohnstein	First Draft
05-22-2017	Christopher Lakin	Edits
05-22-2017	Benjamin Caudill	Edits
05-23-2017	Dwight Hohnstein	Final Draft

TABLE OF CONTENTS

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

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 web 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

Webservers 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 website" anymore, making them even more difficult to protect.

Campaign Objectives

Vulnerability Identification

Rhino Security's 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.

KEY PERSONNEL

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 100, we pride ourselves on both depth and breadth of information security experience.

Chris Lakin - *Cybersecurity Engagement Manager*

Chris Lakin has accumulated over eight years of project management and customer engagement experience across a multitude of industries. A proponent of constant iteration and improvement, his knowledge from time spent in business and marketing adds a valuable perspective to every cybersecurity engagement. Receiving a Masters of Science in Cybersecurity Engineering from the University of Washington, Mr. Lakin excels at connecting the technical with the goals of the business.

Dwight Hohnstein - *Security Consultant*

Dwight Hohnstein started his professional career as a web developer, but quickly found a penchant and passion for application security. His research involves enterprise level applications, firmware modification and reversing, and creating exploits for capture the flag competitions. For examples of his research, you can find the two-part blog post that revealed six high level vulnerabilities in Unitrends Enterprise Backup or Pentesting in AWS Environments on the Rhino Security research section of the website.

Benjamin Caudill - *CEO and Founder*

Benjamin Caudill is an adept cybersecurity professional, researcher, and entrepreneur. A veteran of the defense and aerospace industry, Mr. Caudill led investigations into advanced cyberattacks, coordinating with federal intelligence communities on complex engagements. As Founder and CEO of Rhino Security Labs, Mr. Caudill has built the boutique security firm and turned it into a major player in the penetration testing market. In addition to his executive role, Mr. Caudill oversees company research and development, ensuring the continued development of key offensive technologies.

MOBILE APPLICATION PENETRATION TESTING METHODOLOGY

At Rhino Security Labs, our application penetration testing targets the entire range of vulnerabilities in your Mobile Application or API. Using the same techniques as sophisticated real-world attackers, we providing unique visibility into security risks automated tools often miss. To ensure high quality, repeatable engagements, our penetration testing methodology follows these steps:

1

Reconnaissance

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.

As with malicious hackers, each penetration test begins with information gathering. Collecting, parsing, and correlation information on the target is key to identifying vulnerabilities.

2

Vulnerability Detection

Once the target has been fully enumerated, Rhino Security Labs uses both vulnerability scanning tools and manual analysis to identify security flaws. With decades of experience and custom-built tools, our security engineers find weaknesses most automated scanners miss.

3

Attack and Post-Exploitation

At this stage of the assessment, our consultants review all previous data to identify and safely exploit identified application vulnerabilities. Once sensitive access has been obtained, the focus turns to escalation and movement to identify technical risk and total business impact.

During each phase of the compromise, we keep client stakeholders informed of testing progress, ensuring asset safety and stability.

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

Remediation (Optional)

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.

EXECUTIVE SUMMARY OF FINDINGS

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

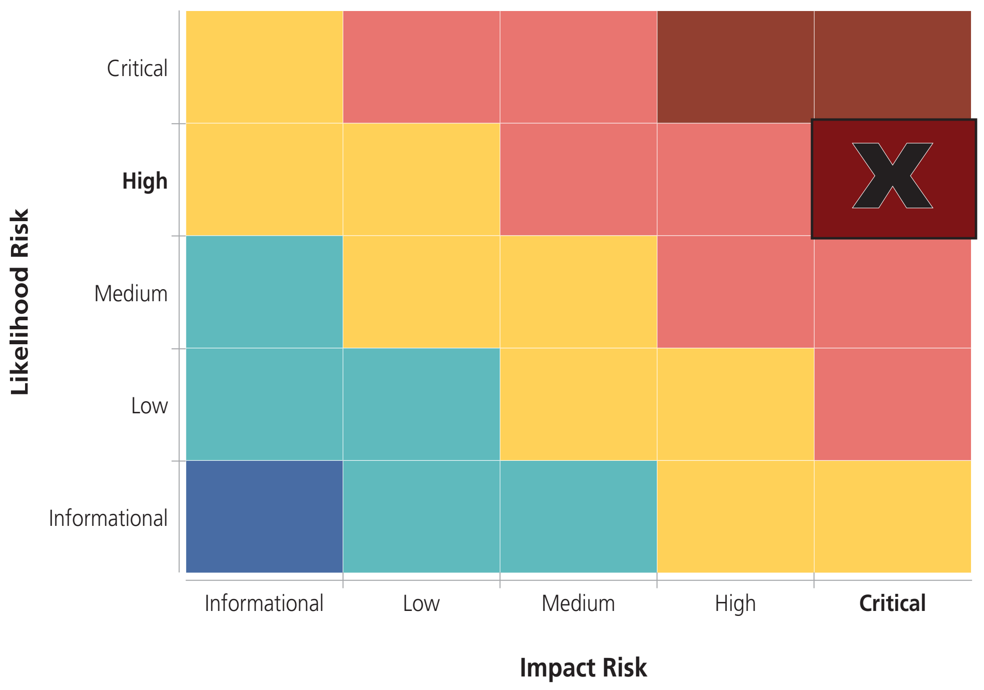
Rhino Security Labs reviewed the security of IniTech's infrastructure and had determined a Critical risk of compromise from external attackers, as shown by the presence of the vulnerabilities detailed in this report.

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 exploit) and Potential Impact (Technical Controls).

Overall Risk Rating: CRITICAL



Summary of Strengths

Rhino Security Labs acknowledged a number of security and technical controls that blocked attempts to carry out malicious actions. 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 IniTech's environment.

- Client implements proper Two-Factor Authentication for login forms.
- Mobile application is using basic encryption to help protect user data storage and in transit.

Summary of Weaknesses

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

- Poor sanitation of data from majority of user fields in application.
- Input fields are not protected against a variety of DDoS and brute forcing attacks.
- Vulnerable to cross site scripting attacks.
- Session tokens and cookies are poorly configured and managed insecurely.

Strategic Recommendations

Not all security weaknesses are technical in nature, nor can they all be remediated by security personnel. Companies often 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. Ensure proper development and repository practices by in-house and 3rd party developers
2. Remove all sensitive pages and directories from publicly accessible servers; Require authentication on such pages as necessary.
3. Remove legacy servers, subdomains, pages, and other web resources when no longer in use
4. Sanitize all user inputs before processing on webserver or other internal resources
5. Enhance security defenses with additional detection and response capabilities, such as a SIEM

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

In addition, the following limitations were put into place:

- Vulnerabilities which would cause outages or interrupt the client's environment were noted but not validated.
- Penetration testing was limited to the agreed upon engagement period, scope, and other additional boundaries set in the contract and service agreement.

Scope of Service

The predetermined scope for Rhino Security Labs to carry out the Mobile application Penetration Test was:

Acme App

Application Platform(s)

iOS / Android

Language(s)

Swift (iOS) / Java (Android)

Application Description

Acme app is a music app that serves 100+ million people from all over the world. Users pay a monthly fee for the service and can listen to listen to unlimited ad-free music. They have come to Rhino Security Labs to make sure that that the implementation of the app and its services are secure in anticipation of their projected growth and the increased attention that will be directed at them.

App Sensitive Assets

Acme app has several sensitive components to their application. The service must utilize a database to store authentication information and payment info and interface with an API to access the data. This data is often the prime target for criminals. If one security hole exists, that could all be that is needed for an attacker to eventually compromise the system and extract sensitive data.

SUMMARY VULNERABILITY OVERVIEW

Rhino Security Labs performed a Mobile application Penetration Test for IniTech, Inc. (IniTech) on 07-01-2017 - 07-31-2017. This assessment utilized both commercial and proprietary tools for the initial mapping and reconnaissance of the site, 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 very high threat to a company's data, and should be fixed on a top-priority basis. They can allow a hacker to completely compromise the environment or cause other serious impacts to the security of the application
HIGH	High severity vulnerabilities should be considered a top priority regarding mitigation. These are the most severe issues and generally, cause an immediate security concern to the enterprise
MEDIUM	Medium severity vulnerabilities are a lower priority, but should still be remediated promptly. These are moderate exploits that have less of an impact on the environment.
LOW	Low severity vulnerabilities are real but trivially impactful to the environment. These should only be remediated after the HIGH and MEDIUM vulnerabilities are resolved.
INFORMATIONAL	Informational vulnerabilities have no impact as such to the environment by themselves. However, they might provide an attacker with information to exploit other vulnerabilities.

VULNERABILITIES BY RISK RATING

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.

2	3	2	3	3
Critical	**High**	**Medium**	**Low**	**Informational**

Total Vulnerabilities Discovered: 13

C1 - Password Brute Force Vulnerability
Risk: CRITICAL

Remediation
Implement password rate-limiting and lockout functionality after three incorrect password attempts.

C2 - SQL Injection via IOS App
Risk: CRITICAL

Remediation
Thoroughly sanitize user inputs or use parameterized queries.

H1 - No Jailbreak Check On iOS Application
Risk: HIGH

Remediation
Before app runtime, implement jailbreak detection by checking for access to root files.

H2 - No Root Check On Android Application
Risk: HIGH

Remediation
Implement root detection before beginning the runtime of your application.

H3 - App Certificate Pinning Not Enforced
Risk: HIGH

Remediation
Bundle the server’s SSL certificate with the application and enforce SSL pinning.

M1 - Sensitive App Data Can Be Backed up via Android DB Backup
Risk: MEDIUM

Remediation
Set the flag android:allowBackup to false on the manifest file.

M2 - Insecure Android WebView Implementation Risk: **MEDIUM**

Remediation

Use “android.webkit.webview” which has built in security functions to prevent against code execution.

L1 - Android App Logs Sensitive System Information in Temporary Files Risk: **LOW**

Remediation

Avoid storing sensitive information in plaintext. Implement file decryption via user’s password.

L2 - SSL Cookie without Secure Flag Set Risk: **LOW**

Remediation

Ensure all cookies issued have the secure flag set, preventing plaintext transmission (HTTP).

L3 - iOS Application Uses 'malloc' Function Call Risk: **LOW**

Remediation

Use calloc() to replace malloc() where possible, as calloc() zero initializes the memory buffer.

I1 - HTML Does Not Specify Charset Risk: **INFORMATIONAL**

Remediation

Ensure that all pages have the content header of a widely recognized character set such as UTF-8.

I2 - User Agent Dependent HTTP Response Risk: **INFORMATIONAL**

Remediation

Ensure that the alternate applications are thoroughly tested or consolidate to one application for all User-Agents.

I3 - Cacheable HTTPS Response Risk: **INFORMATIONAL**

Remediation

Ensure that all pages have the content headers for Cache-Control and Pragma set to no-cache.

VULNERABILITY FINDINGS

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

C1 Password Brute Force Vulnerability

Risk Rating: CRITICAL

Exploitation Likelihood: **CRITICAL**
Potential Impact: **CRITICAL**

Description

A password brute force vulnerability would allow an attacker to attempt as many passwords as necessary to eventually recover the password. This would then allow the attacker to gain access to the account and retrieve sensitive data inside.

While brute force attacks can be easy to identify with the right access logs, the impact on a vulnerable application can be critical due to the potentially compromised access of a user's account.

Affected Application

Acme Mobile App

Affected Platform(s)

iOS/Android Apps

Remediation

To prevent this vulnerability, implement rate limiting and a lockout functionality. For example, if a pre-determined limit of password attempts (3) per X (1 minute) amount of time is exceeded, credential entering functionality should be denied for increasingly greater amounts of time eg. 1 min. 2 min. 4 min. Additionally, including multifactor authentication would slow an attacker from successfully logging in should they acquire the password.

Testing Process

Twenty authentication attempts were performed in the span of one minute without causing a lockout, indicating that brute forcing is possible. Further brute force testing shows that no rate-limit exists and the only bottleneck is the app and device's processing speed.

C2 SQL Injection via IOS App

Risk Rating: CRITICAL

Exploitation Likelihood: **CRITICAL**
Potential Impact: **CRITICAL**

Description

The Input to the application is not properly sanitized and parameterized queries are not used. Lacking these measures could lead to a SQL injection on a local or remote database which could result in complete compromise of the application. Once a SQL injection is identified it can be rather trivial to exploit with tools such as Sqlmap.

Affected Application

Acme Mobile App

Affected Platform(s)

iOS

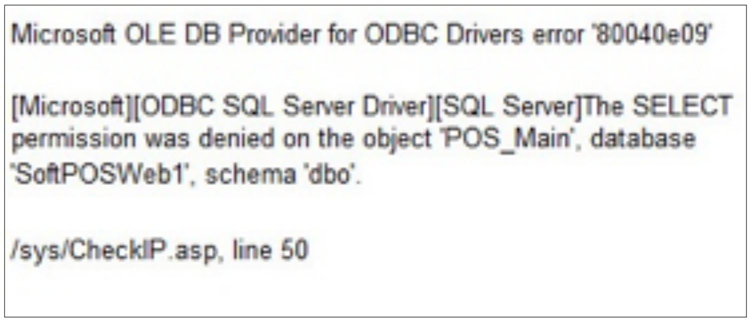
Remediation

When possible use prepared statements with parametrized queries, they can at times impact performance. If that is the case, fully sanitize inputs or create a whitelist for user input.

More information on defending against SQL injection here:
https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

Testing Process

Manual SQL injection was performed and the modification of the database was successful. This allowed the addition of a user or changing of the password to name a few possibilities.



```
Microsoft OLE DB Provider for ODBC Drivers error '80040e09'

[Microsoft][ODBC SQL Server Driver][SQL Server]The SELECT
permission was denied on the object 'POS_Main', database
'SoftPOSWeb1', schema 'dbo'.

/sys/CheckIP.asp, line 50
```

H1 No Jailbreak Check On iOS Application

Risk Rating: HIGH

Exploitation Likelihood: **HIGH**
Potential Impact: **CRITICAL**

Description

No jailbreak detection has been implemented in the application. This allows a user to modify an app on a jailbroken device, inducing behaviors that otherwise would not occur. Examples include removing ASLR, overwriting critical functions, and more. This would leave the application more vulnerable and allow an attacker to more easily exploit it.

Affected Application

Acme Mobile App

Affected Platform(s)

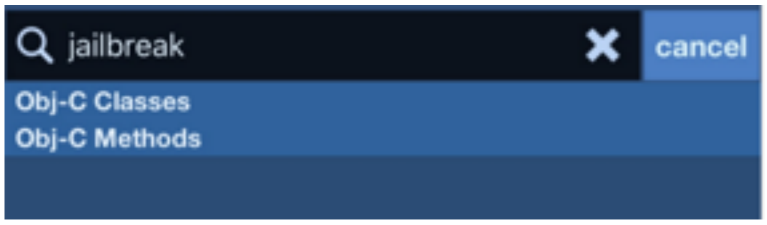
iOS

Remediation

Implement jailbreak detection before beginning the runtime of your application, such as seeing if you can access root files of the device.

Testing Process

This vulnerability was discovered by running the application on a jailbroken iOS device. In the screenshot below, it is seen that there are no Classes or Methods which implement jailbreak checks.



H2 No Root Check On Android Application

Risk Rating: HIGH

Exploitation Likelihood: **HIGH**
Potential Impact: **CRITICAL**

Description

No detection is implemented in the application to determine whether the Android device has been 'rooted'. This allows a user to modify an app on a rooted Android device, inducing behaviors that otherwise would not occur.

Examples include removing ASLR, overwriting critical functions, and more. This would leave the application more vulnerable and allow an attacker to more easily exploit it.

Affected Application

Acme Mobile App

Affected Platform(s)

Android Apps

Remediation

Implement 'root' detection before beginning the runtime of your application, such as looking for the presence of files and packages specific to a 'rooted' device.

Testing Process

This was discovered by running the app on both normal and rooted Android devices.

H3 App Certificate Pinning Not Enforced

Risk Rating: HIGH

Exploitation Likelihood: **CRITICAL**
Potential Impact: **HIGH**

Description

SSL Pinning makes sure that the client checks the server's certificate against a known copy of that certificate. If the server's SSL certificate is bundled inside the application, then an SSL request first validates that the server's certificate exactly matches the bundle's certificate.

When pinning is used, even when using a proxy with an imported certificate, an attacker will not be able to see sensitive data. This occurs because the imported certificate does not match the certificate that was bundle/pinned with the application.

Affected Application

Acme Mobile App

Affected Platform(s)

iOS/Android Apps

Remediation

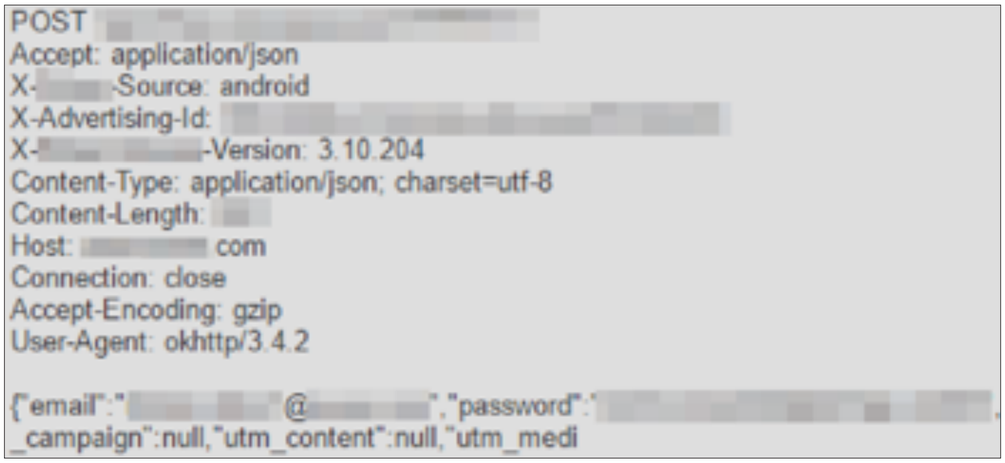
Bundle the server's SSL certificate with the application and enforce SSL pinning to ensure that the certificate is validated before establishing a secure connection. This will make it more difficult for a successful man-in-the-middle attack to occur.

For more information visit:
https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning#iOS

Testing Process

A proxy certificate was used to tunnel the application data through a monitoring proxy. This was succesful because app certificate pinning was not enable.

The following is a recorded login sequence from a user accessing the app while there was an active man-in-the-middle between the app and server:



```
POST [redacted]
Accept: application/json
X-[redacted]-Source: android
X-Advertising-Id: [redacted]
X-[redacted]-Version: 3.10.204
Content-Type: application/json; charset=utf-8
Content-Length: [redacted]
Host: [redacted] com
Connection: close
Accept-Encoding: gzip
User-Agent: okhttp/3.4.2

{"email":"[redacted]@[redacted]","password":"[redacted]
_campaign":null,"utm_content":null,"utm_medi
```

M1 Sensitive App Data Can Be Backed up via Android DB Backup

Risk Rating: MEDIUM

Exploitation Likelihood: **MEDIUM**
Potential Impact: **MEDIUM**

Description

The flag [android:allowBackup] should be set to false. By default it is set to true and allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device. This allows for the viewing of log files for example, which can contain sensitive information such as user data.

Affected Application

Acme Mobile App

Affected Platform(s)

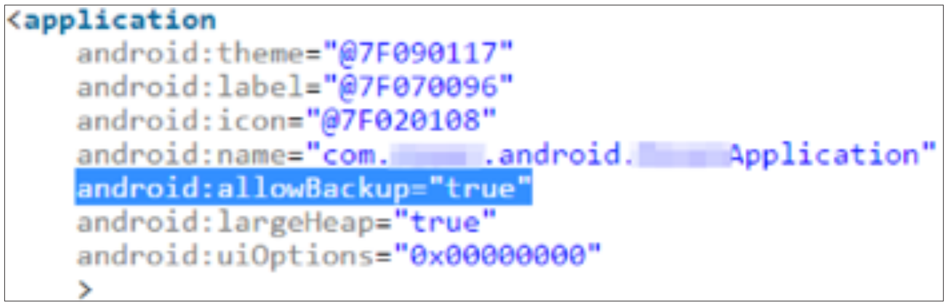
Android Apps

Remediation

Set the flag android:allowBackup to false on the manifest file to disallow the backup functionality. This will make it much more difficult to access sensitive data.

Testing Process

The vulnerability was detected by checking the application Manifest.xml.



```
<application
    android:theme="@7F090117"
    android:label="@7F070096"
    android:icon="@7F020108"
    android:name="com.[redacted].android.[redacted]Application"
    android:allowBackup="true"
    android:largeHeap="true"
    android:uiOptions="0x00000000"
    >
```

M2 Insecure Android WebView Implementation

Risk Rating: MEDIUM

Exploitation Likelihood: **MEDIUM**
Potential Impact: **MEDIUM**

Description

User controlled code is executable within the implemented WebView. This allows for execution of JavaScript which can result in dangerous operations like system-wide keylogging. While this attack requires more of a targeted approach, for the average attacker it is still quite possible.

Affected Application

Acme Mobile App

Affected Platform(s)

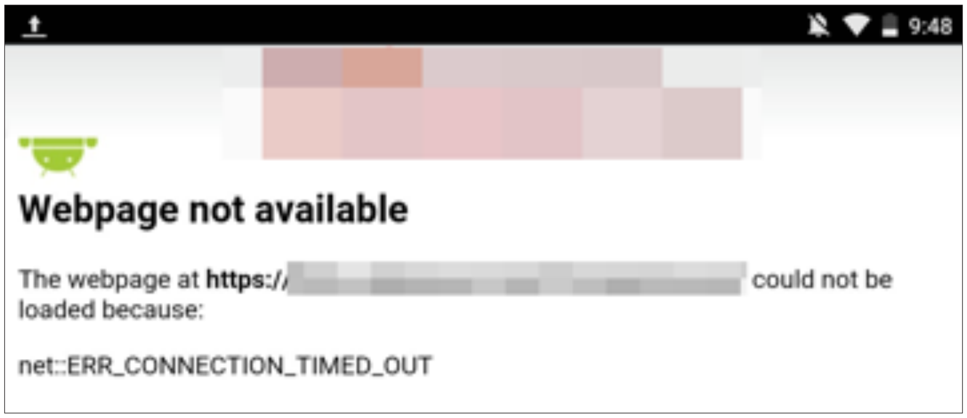
Android

Remediation

Use "android.webkit.webview" which has built in security functions to prevent against code execution. Bolts WebView was detected in this application and should not be used.

Testing Process

his was detected from the Mobile Security Framework and confirmed manually by the assesor. The image below shows the use of an insecure implementation of Android web view.



Webpage not available

The webpage at **https://**[redacted] could not be loaded because:

net::ERR_CONNECTION_TIMED_OUT

L1 Android App Logs Sensitive System Information in Temporary Files

Risk Rating: LOW

Exploitation Likelihood: **LOW**
Potential Impact: **LOW**

Description

The application was found to store sensitive information, such as IMEI numbers, user-data, cookies, passwords and more in clear text. With this data an attacker could take over the users account and impersonate them.

Affected Application

Acme Mobile App

Affected Platform(s)

Android Apps

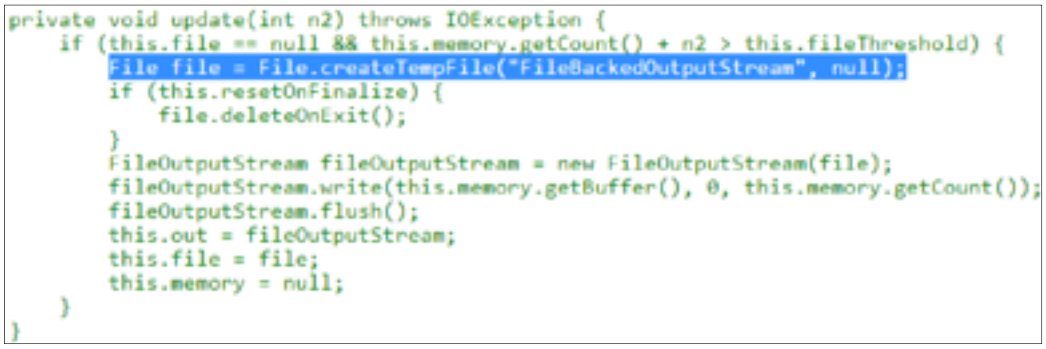
Remediation

Implement encryption where the user's password decrypts the files, instead of storing the data in plain text. This will make it more difficult for an attacker via a malicious app or some other means, extract sensitive pieces of data.

Testing Process

The vulnerability was detected by looking at the application source code.

The temporary file creation is shown in the screenshot here:



```
private void update(int n2) throws IOException {
    if (this.file == null && this.memory.getCount() + n2 > this.fileThreshold) {
        File file = File.createTempFile("FileBackedOutputStream", null);
        if (this.resetOnFinalize) {
            file.deleteOnExit();
        }
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        fileOutputStream.write(this.memory.getBuffer(), 0, this.memory.getCount());
        fileOutputStream.flush();
        this.out = fileOutputStream;
        this.file = file;
        this.memory = null;
    }
}
```

L2 SSL Cookie without Secure Flag Set

Risk Rating: LOW

Exploitation Likelihood: **LOW**
Potential Impact: **LOW**

Description

Several cookies (which appear to be session tokens) were issued by the application which do not have the secure flag set.

The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text.

To accomplish this goal, browsers which support the secure flag will only send cookies with the secure flag when the request is going to a HTTPS page. Said in another way, the browser will not send a cookie with the secure flag set over an unencrypted HTTP request. By setting the secure flag, the browser will prevent the transmission of a cookie over an unencrypted channel.

Proof-of-concept tools such as "Firesheep" have reduced exploitation of insecure cookies to a simple point-and-click.

Affected Application

Acme Mobile App

Affected Platform(s)

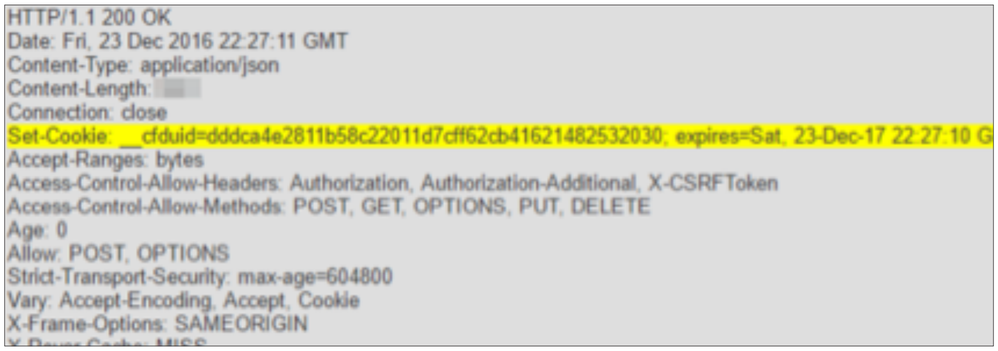
iOS/Android Apps

Remediation

Ensure all cookies issued have the secure flag set, preventing plaintext transmission (HTTP).

Testing Process

This vulnerability was detected by automated scanning. As seen in the HTTP Response, the "Secure" flag is missing:



```
HTTP/1.1 200 OK
Date: Fri, 23 Dec 2016 22:27:11 GMT
Content-Type: application/json
Content-Length: [redacted]
Connection: close
Set-Cookie: __cfduid=dddca4e2811b58c22011d7cff62cb41621482532030; expires=Sat, 23-Dec-17 22:27:10 G
Accept-Ranges: bytes
Access-Control-Allow-Headers: Authorization, Authorization-Additional, X-CSRFToken
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Age: 0
Allow: POST, OPTIONS
Strict-Transport-Security: max-age=604800
Vary: Accept-Encoding, Accept, Cookie
X-Frame-Options: SAMEORIGIN
```

L3 iOS Application Uses 'malloc' Function Call

Risk Rating: LOW

Exploitation Likelihood: **LOW**
Potential Impact: **LOW**

Description

he application uses malloc() instead of calloc(), which could cause memory corruption or cause the application to read unexpected data. This could result in

Affected Application

Acme Mobile App

Affected Platform(s)

iOS

Remediation

Use calloc() to replace malloc() where possible, as calloc() zero initializes the memory buffer.

Testing Process

This was detected from the Mobile Security Framework and confirmed by the assessor.

I1 HTML Does Not Specify Charset

Risk Rating: INFORMATIONAL

Exploitation Likelihood: **INFORMATIONAL**
Potential Impact: **INFORMATIONAL**

Description

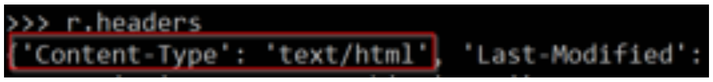
The response HTML does not specify a charset which could cause the browser to interpret data it receives as something other than what was intended. If the browser interprets the coding to be UTF-7, XSS could be possible.

Remediation

Ensure that all pages have the content header of a widely recognized character set such as UTF-8 and that it is the first line in the header. When the browser parses the html and hits the meta tag with the character set, it will start over again with that character set in mind.

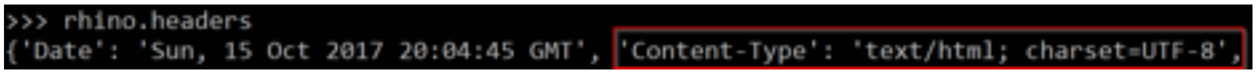
Testing Process

The vulnerability was identified by an automated scanner and confirmed by the assessor. In the first image below, you can see the absence of a specified charset. In the second image, the charset is correctly specified. To test this, either execute curl -v url or use python requests and list the headers by calling request_object.headers.



```
>>> r.headers
{'Content-Type': 'text/html', 'Last-Modified':
```

An example of a specified charset.



```
>>> rhino.headers
{'Date': 'Sun, 15 Oct 2017 20:04:45 GMT', 'Content-Type': 'text/html; charset=UTF-8',
```

I2 User Agent Dependent HTTP Response

Risk Rating: INFORMATIONAL

Exploitation Likelihood: **INFORMATIONAL**
Potential Impact: **MEDIUM**

Description

The response from the web application is dependent on the User-Agent header. This signifies that there are other attack surfaces besides the main web page served to desktop/laptop users for example. An attacker could instead attack the mobile version of the web application which historically is less thoroughly tested and relies on weaker security mechanisms.

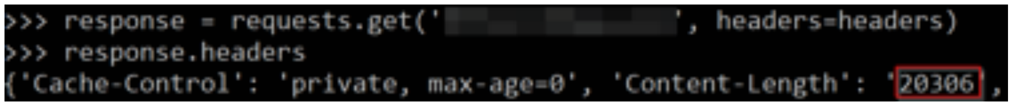
Remediation

Ensure that the alternate applications are thoroughly tested or consolidate to one application for all User-Agents.

Testing Process

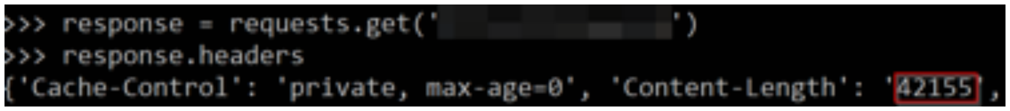
The vulnerability was identified by sending different User-Agent strings and getting different results back from the application. The first image (mobile user-agent) has content length of around half of the second image indicating that there are at least two different versions of the application.

Mobile:



```
>>> response = requests.get('[redacted]', headers=headers)
>>> response.headers
{'Cache-Control': 'private, max-age=0', 'Content-Length': '20306',
```

Desktop/Laptop:



```
>>> response = requests.get('[redacted]')
>>> response.headers
{'Cache-Control': 'private, max-age=0', 'Content-Length': '42155',
```

13 Cacheable HTTPS Response

Risk Rating: INFORMATIONAL

Exploitation Likelihood: **INFORMATIONAL**
Potential Impact: **INFORMATIONAL**

Description

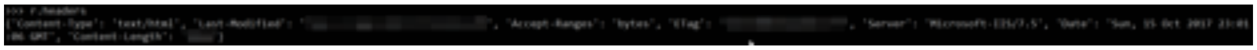
Information returned by the web server can be cached. Other users of the computer could retrieve the cached data and if it was sensitive, such as credit card numbers, SSN or usernames they could leverage that information for additional attacks.

Remediation

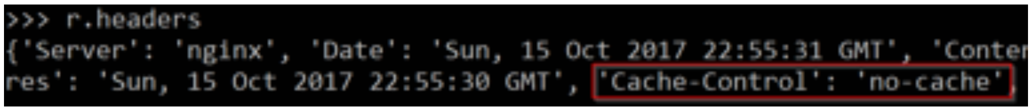
Ensure that all pages have the content header for Cache-Control and Pragma set to no-cache. This can be done by adding 'no-cache' in the meta tag for each page.

Testing Process

The vulnerability was identified by an automated scanner. By looking at the server's headers in the first image there is no Cache-Control value specified. In the second image, proper cache control is performed by specifying 'no-cache'. To view the headers execute curl -v url or use python requests and execute requests_object.headers.



```
>>> r.headers
{'Content-Type': 'text/html', 'Last-Modified': '[redacted]', 'Accept-Ranges': 'bytes', 'ETag': '[redacted]', 'Server': 'Microsoft-IIS/7.5', 'Date': 'Sun, 15 Oct 2017 23:01
:06 GMT', 'Content-Length': '[redacted]'}
```



```
>>> r.headers
{'Server': 'nginx', 'Date': 'Sun, 15 Oct 2017 22:55:31 GMT', 'Conter
res': 'Sun, 15 Oct 2017 22:55:30 GMT', 'Cache-Control': 'no-cache'
```

APPENDIX A: TOOLS & SCRIPTS

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 social engineering methodology. Below is the toolset our consultants use during a social engineering assessment.

Mobile App Tools

Burpsuite Professional

Burpsuite 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 identified vulnerabilities. It can also be used to analyze the data going between mobile applications and their servers.

Needle

eedle is an open source, modular framework to streamline the process of conducting security assessments of iOS apps including Binary Analysis, Static Code Analysis, Runtime Manipulation using Cycript and Frida hooking, and so on.

Cycript

Cycript allows developers to explore and modify running iOS applications using a hybrid of Objective-C++ and JavaScript syntax through an interactive console. Cycript is used heavily in the analysis and mapping of iOS apps.

iRET (iOS Reverse Engineering Toolkit)

iRET is an open source tool used to analyze and evaluate iOS applications. The toolkit can perform binary analysis, keychain analysis, database analysis, log viewer, Plist viewer, and more.

Introspy-Android

Introspy-Android can be installed on a rooted device and dynamically configured to hook security-sensitive Android APIs at run-time. The tool records all the relevant API calls made by an application, including function calls, arguments and return values. It then performs tests for security issues in real time and persists the results in a database and in the Android logging system

Android-SSL-TrustKiller

Blackbox tool to bypass SSL Certificate pinning for most applications running on an Android device.

drozer

drozer (formerly Mercury) is the leading security testing framework for Android. drozer allows you to search for security vulnerabilities in apps and devices by assuming the role of an app and interacting with the Dalvik VM, other apps' IPC endpoints and the underlying OS.

Custom Scripts and application

In addition to the above tools, Rhino Security Labs also makes use of its own proprietary tools and scripts to quickly adapt to new and unique environments.

APPENDIX B: LIST OF CHANGES MADE TO INITECH SYSTEMS

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.

Affected Area	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.

APPENDIX C: OWASP MOBILE TOP 10

The Open Mobile application Security Project surveys security professionals across the globe collecting information on the most frequently found security vulnerabilities within mobile applications. The results are compiled, categorized, and ranked to determine the top ten. Rhino Security Labs uses both automated and manual processes to identify if these vulnerabilities are present within your application. The appendix is to help explain how some of your vulnerabiltiies might be categorized.

M1-Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.
M2-Insecure Data Storage	This new category is a combination of M2 + M4 from Mobile Top Ten 2014. This covers insecure data storage and unintended data leakage.
M3-Insecure Communication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
M4-Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: • Failing to identify the user at all when that should be required • Failure to maintain the user's identity when it is required • Weaknesses in session management
M5-Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasn't done correctly.

M6-Insecure Authorization

This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.).

If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.

M7-Poor Code Quality

This was the "Security Decisions Via Untrusted Inputs", one of our lesser-used categories. This would be the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.

M8-Code Tampering

This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification.

Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.

M9-Reverse Engineering

This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.

M10-Extraneous Functionality

Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.

888.944.8679

info@rhinosecuritylabs.com

1200 East Pike Street Suite 510 | Seattle, WA 98122

RHINO SECURITY LABS