andersenlab.com

Security Assessment Report

Sample Application Security Assessment

HIPAA COMPLIANT
GDPR
aws CERTIFIED Solutions Architect Associate
ISO 9001
ISO/IEC 27001
ISO 13485
IAOP 100 2024 The 2024 Global Outsourcing 100
TOP 1000 2024 FINANCIAL TIMES Europe's fastest-growing companies
TISAX®

Independent Security Assessment Report

Additionally, it is worth **emphasizing** that the findings and remediation recommendations are the result of a point-in-time assessment based on the state of the Client environment as of March 15, 2023. Andersen therefore does not provide any assurance related to configuration or control modifications in the Client environment, changes in regulatory or compliance requirements, discoveries of new vulnerabilities and attack techniques, or any other future event that may impact the Client's security posture.

The information contained in this report represents a fair and unbiased assessment of the Client's environment based on the agreed upon criteria as defined in the **Statement of Work**. This report is provided to the Client as notification of outstanding security risks that threaten the confidentiality, integrity, and availability of sensitive information, as well as to provide assistance and direction with remediation. The evidence and references provided for each finding serve as the basis for our qualified opinions in this report.

Andersen has provided this report solely for private and internal use by the Client, and it may not be shared or redistributed without Andersen's express written consent. Andersen's assessments focus exclusively on information security and the conclusions arrived at in this report should not be considered to be a representation or endorsement of the Client's products or services.

Andersen has performed the Application Security Assessment for ABC Soft, ("Client") while acting as an independent security assessor. This assessment was performed with the intent of evaluating security, and resiliency of Client's Sample IT systems.

The methodology utilized during this assessment is detailed in **Methodology**. Andersen developed this methodology based on extensive professional experience and information system security assessment best practices gathered from the NIST Risk Management Framework, Open Source Security Testing Methodology Manual ("OSSTMM"), the National Institute of Standards and Technology ("NIST") Special Publication 800-115: Technical Guide to Information Security Testing and Assessment, the Penetration Testing Execution Standard ("PTES"), NIST Guide Details Forensic Practices, various CIS Benchmarks, and the Open Web Application Security Project ("OWASP") Testing Guide.

While this type of assessment is intended to mimic a real-world attack scenario or identify the capacity of the existing controls, Andersen is bound by **rules-of-engagement, defined scope, allocated time, and additional related constraints**. Andersen has made every effort to perform a thorough and comprehensive analysis and to provide appropriate remedial advice. However, inherit limitations, errors, misrepresentations, and changes to the Client environment may have prevented Andersen from identifying every security issue that was present in the Client environment at the time of testing. Therefore, the findings included in this report should be considered to be representative of what a similarly skilled attacker could achieve with comparable resources, constraints, and time frame.

Management Summary

This report details the findings of the Sample Application Security Assessment carried out between **March 02, 2024 and March 15, 2023.**

The most important objective of the assessment was to determine whether and how a malicious user can gain unauthorized access to assets that affect the fundamental security of the system, files and data, and confirm that the applicable controls required by ABC Soft are in place.

The security team has conducted the assessment based on the **Application Security Assessment methodology**.

The following issues are evaluated as Critical or High risks and require immediate attention and remediation:

Account Takeover

through weak Reset Password functionality

Sensitive Information

hardcoded in application code

Insecure Direct Object References

users can modify other user's data (IDOR)

Andersen identified numerous Medium risk issues that address failures to adhere to established security best practices. In some cases, these vulnerabilities increase the attack surface of the assets and may make the exploitation of other weaknesses easier. These findings should be addressed in turn and as time permits.

The security team recommends that the client should conduct a session for planning the remediation of the identified risks, starting with the most important findings.

RESULTS

As a result of conducting this engagement, Andersen has determined that cumulatively the issues identified pose a **High risk to ABC Soft.** This evaluation was determined by assessing the severity and number of issues identified throughout the environment as well as Andersen's experience in assessing similar systems.

The overall risk can be lowered by remediating the vulnerabilities detailed in the following chapters.

Scope of Work

BACKGROUND INFORMATION

Andersen performed Application Security Assessment to assess the risk that a real life, targeted attacker poses to the security and integrity of the ABC Soft Sample. Understanding the current vulnerabilities is the first step in remediating and ultimately enhancing ABC Soft's overall security maturity.

The purpose of the assignment was to identify and evaluate any risks or potential issues that could impact Confidentiality, Integrity or Availability of the systems in scope. In this assessment, both automated and manual security testing techniques were used in order to identify weakness in the systems in scope from an attacker's perspective.

OBJECTIVES

The objective of this assignment is to help ABC Soft strengthen the security posture against cyber threats.

Securing vulnerabilities and reducing risks within the systems will lead to a drastic reduction in the likelihood of:

- Exploitation of publicly available exploits through lack of patching;
- Financial loss through regulatory penalties;
- Disruption of availability through a lack of rate-limiting techniques;
- Breach of integrity through weak authorization checks;
- Systems compromise, data alteration or data destruction attacks;
- Information theft through poor or non-existent cryptographic controls;

SCOPE OVERVIEW

The scope of the assessment included the following assets as authorized by the ABC Soft:

iOS	android	API
iOS Application	Android Application	API

- Assessment type: **Application Security Assessment**
- Assessment method: **Gray Box**
- Environment: **Staging**

The Application Security Assessment was performed in the dates between **March 02, 2024 and March 15, 2023.**

LIMITATIONS

Denial-of-Service (DoS) testing was not performed during this engagement.

This was a time-boxed security assessment. During a time-boxed engagement, the Cyber Security Team prioritizes assessment of the most sensitive portions and functions of the systems in scope.

No other specific limitations were defined in the scoping phase by the client.

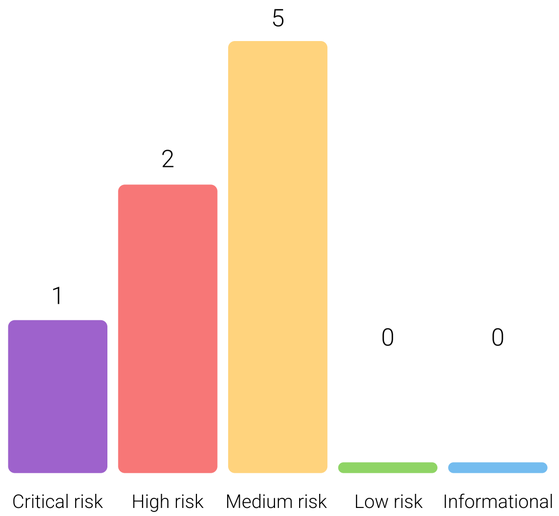
Summary of Findings

Using automated and manual techniques, Andersen identified a total of **8 findings within Sample environment**. These weaknesses threaten the confidentiality, integrity, and availability of the application, the environment, and the data contained within it.

RISK BREAKDOWN

The following table summarizes the quantity and severity of the findings identified during this assessment:

Residual Risk Severity	Total
Critical	1
High	2
Medium	5
Low	0
Informational	0
Total	0



CATEGORY BREAKDOWN

The table below contains a list of areas where vulnerabilities have been identified. The vulnerability categories are defined following the Common Weakness and Enumeration (CWE) DB.

Vulnerability Categories	Total
Broken Authentication	1
Security Misconfiguration	2
Information Exposure	5
Missing Authorization	0
Cleartext Storage	0
Hardening	0

COMPONENT BREAKDOWN

The table below contains a list of affected components where vulnerabilities have been identified.

Vulnerability Categories	Total
Application	3
Mobile	5

Vulnerability Details

For each finding, Andersen uses a composite risk score that takes into account the severity of the risk, application's exposure, technical difficulty of exploitation, and other factors. For an explanation of Andersen's risk rating and vulnerability categorization, see the Methodology section.

The table below lists the vulnerabilities identified during the assessment:

TABLE OF VULNERABILITIES

Residual Risk	CIA Impact	Title	Identifier
Critical	**C I A**	Account Takeover through weak Reset Password functionality	SAM-6
High	**C** I A	Sensitive information hardcoded in application code	SAM-2
High	**C I** A	Insecure Direct Object References: Users can modify other user's data (IDOR)	SAM-7
Medium	**C** I A	Insecure data storage (iOS)	SAM-0
Medium	**C I** A	Insecure data storage in Shared Preferences	SAM-1
Medium	**C** I A	Application has no Jailbreak detection capabilities	SAM-3
Medium	**C** I A	Rooted device detection is easy to bypass	SAM-4
Medium	**C** I A	Application accepts any SSL Certificate (No SSL Pinning)	SAM-5

Vulnerability Details

V1. Account Takeover through weak Reset Password functionality

An attacker is able to hijack user accounts through the weak implementation of Forgot Password functionality. The reset/forgot password functionality is programmed without security considerations.

It is common for an application to have a mechanism that provides a means for a user to gain access to their account in the event they forget their password. Very often the password recovery mechanism is weak, which has the effect of making it more likely that it would be possible for an attacker other than the legitimate system user to gain access to that user's account. Weak password recovery schemes completely undermine a strong password authentication scheme.

Account takeover is the possibility of attackers using different attack vectors, such as broken access controls, insecure direct object reference, broken business logic attacks, session hijacking, or other client-side vectors in order to compromise user credentials.

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Go to the "Forgot password" page
- Submit your username/email through the form
- Inspect the Recover Password URL
- View if it can be used to reset the password of other users

Impact

An attacker can take advantage of the application feature to recover user's accounts forgotten passwords in order to gain access into the accounts with the same privileges as the original user. This means that if the attacker manages to enter as a normal user, he might have limited access to only view some important information. On the other hand, if he manages to enter as an administrative user with global access to the system, he would have almost total control of the application together with its content (with the limitations of the web application in itself).

Affected Entity	Sample · Identifier · SAM-6
Risk Statement	An attacker could gain unauthorized access to other user's account by retrieving legitimate users' authentication credentials using the Forgot Password functionality.
Affected Component	Application Identified Controls: None Identified
Residual Risk	Critical (CVSS Score: 9.8)
Classification	Broken Authentication Likelihood: High
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Location	ResetPassword Activity https://abcsoft.com/account/resetPassword

Vulnerability Details | V1

The following evidence has been gathered to illustrate this vulnerability.

Request

```
GET /api/collab/comments?id=AVL_R13390&since=0&objType=rule
HTTP/2 Host:abcsoft.com
X-Forwarded-Host: https://evil-hacker.com
Cookie:
Sec-Fetch-Site: same-origin
Te: trailersN/A
```

Response

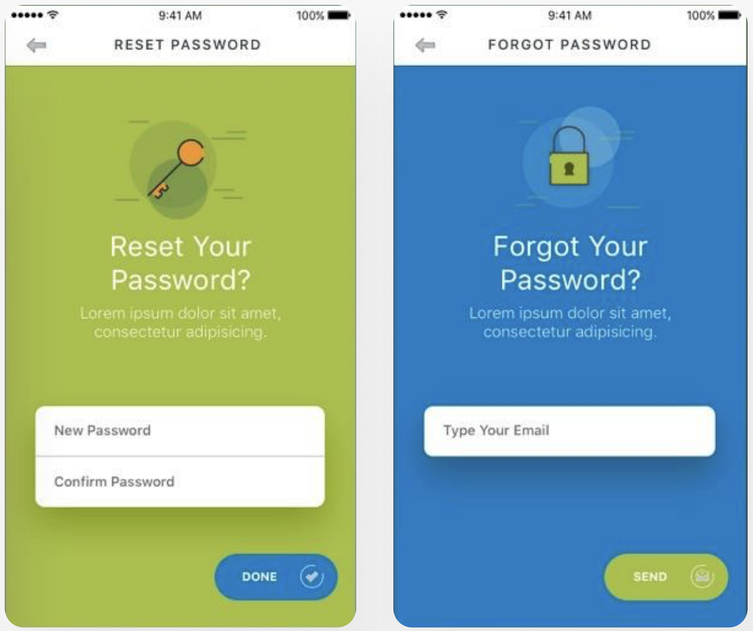
```
HTTP/2 200 OK
Date: Tue, 01 Nov 2022 18:19:40 GMT
Content-Type: application/json
Content-Length: 287
Server: nginx
```

Recommendations

Easy

- Make sure that all input supplied by the user to the password recovery mechanism is thoroughly filtered and validated.
- Do not use standard weak security questions and use several security questions.
- Make sure that there is throttling on the number of incorrect answers to a security question. Disable the password recovery functionality after a certain (small) number of incorrect guesses.
- Require that the user properly answers the security question prior to resetting their password and sending the new password to the e-mail address of record.
- Never allow the user to control what e-mail address the new password will be sent to in the password recovery mechanism.
- Assign a new temporary, totally random, and unpredictable password rather than revealing the original password.

- https://cwe.mitre.org/data/definitions/840.html
- https://cwe.mitre.org/data/definitions/287.html
- https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sh eet.html



Example POC

Vulnerability Details

V2. Sensitive information hardcoded in application code

Security relevant information is found hardcoded within the application source code. Using tools designed to parse the code, sensitive data can be detected as hardcoded within the application source code. If the application uses hard- coded constants instead of dynamic content generation for security important values, they are disclosed to potential attackers.

Protecting authentication tokens, private information, and other sensitive data is key to application security. Public data should be available to everyone, but sensitive and private data must be protected.

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Inspect the application source code
- View the hardcoded security information

Impact

Hard coding sensitive information, exposes it to attackers. Disclosing sensitive information in hardcoded constants increases the area of attack because the information can be used to discover further functionalities and more dangerous vulnerabilities.

Affected Entity	Sample · Identifier · SAM-2
Risk Statement	An attacker will have access to sensitive information hardcoded within the application code and conduct further attacks
Affected Component	Application Identified Controls: None Identified
Residual Risk	High (CVSS Score: 7.5)
Classification	Information Exposure Likelihood: High
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	Android Application

Vulnerability Details | V2

```
public static SignUpResponse mock() {
    SignUpResponse signUpResponse = new SignUpResponse();
    signUpResponse.status = "OK";
    signUpResponse.ssoId = "234432";
    signUpResponse.firstName = "Mock";
    signUpResponse.lastName = "Doe";
    signUpResponse.email = "mock.doe@test.com";
    signUpResponse.msisdn = "07444zzzzzz";
    signUpResponse.username = "mock.doe";
    return signUpResponse;
}
```

Test credentials found within application source code

Recommendations

Moderate

- A general recommendation is to use the dynamic generation of variables that contain sensitive information. Do not hardcode sensitive data into the application code, if the code is accessible by external users.
- Use obfuscation techniques to make the code harder to read and understand.
- Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs. Apply controls as per the classification.

- https://cwe.mitre.org/data/definitions/547.html
- https://owasp.org/www-project-top-ten/2017/A3_2017- Sensitive_Data_Exposure
- https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d- Testing-Data-Storage.md

Vulnerability Details

V3. Insecure Direct Object References: Users can modify other user's data (IDOR)

Internal users are able to access and modify other users' data due to the application being vulnerable to IDOR.

Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access or modify objects directly. Attackers can use direct references

to restricted resources and the application fails to verify the user is authorized to access the exact resource.

IDOR brings, depending on the format/pattern in place, a capacity for the attacker to mount an enumeration attack in order to try to probe access to the associated objects. An enumeration attack can be described in the way in which the attacker builds a collection of valid identifiers using the discovered format/pattern and test them against the application.

The most basic IDOR scenario happens when the application references objects using easy to guess IDs. For example, they can be incremental integers, they can contain predictable words like the email of the user, or a folder name.

Suppose that an application lets you modify your data on the following endpoint:

```
POST /api/users/78963
{some data}
```

In this simple scenario, all you have to do is substitute your ID, which is 78963, with another guessed value and modify that user's data

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Login to the Application
- In the Affected Assets: increment the ID to access the resources of other users
- Submit the request
- View if the data is modified

Affected Entity	Sample · Identifier · SAM-7
Risk Statement	Internal users are able to modify other user's data and have unauthorized access to resources or operations
Affected Component	Application Identified Controls: None Identified
Residual Risk	High (CVSS Score: 8.7)
Classification	Missing Authentication Likelihood: High
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:N
Location	ResetPassword Activity https://abcsoft.com/account/id

Vulnerability Details | V3

Impact

IDOR vulnerabilities are most commonly associated with horizontal privilege escalation, but they can also arise in relation to vertical privilege escalation. An attacker might be able to perform horizontal and vertical privilege escalation by altering the user to one with additional privileges while bypassing access controls. Other possibilities include exploiting password leakage or modifying parameters once the attacker has landed on the user's accounts page, for example.

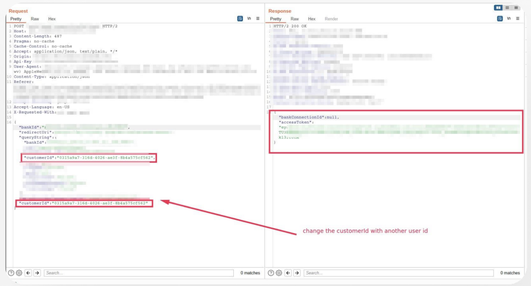
The following evidence has been gathered to illustrate this vulnerability.

Request

```
GET /api/collab/comments?id=AVL_R13390&since=0&objType=rule HTTP/2
Host:abcsoft.com
X-Forwarded-Host: https://evil-hacker.com Cookie:
Sec-Fetch-Site: same-origin
Te: trailersN/A
```

Response

```
HTTP/2 200 OK
Date: Tue, 01 Nov 2022 18:19:40 GMT
Content-Type: application/json
Content-Length: 287
Server: nginx
```



Example POC

Recommendations

Moderate

Preventing insecure direct object references requires selecting an approach for protecting each user-accessible object (e.g., object number, filename):

- **Use per user or session indirect object references.** This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource's database key, a drop-down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server. OWASP's ESAPI includes both sequential and random access reference maps that developers can use to eliminate direct object references.
- **Check access.** Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.
- https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Ref erence_Prevention_Cheat_Sheet.html
- https://portswigger.net/web-security/access-control/idor
- https://www.netsparker.com/blog/web-security/insecure-direct-object- reference-vulnerabilities-idor/

Vulnerability Details

V4. Insecure data storage (iOS)

Sensitive data is stored unencrypted within the application folder.

It may be possible for an attacker to retrieve the content of the files using various methods. Because the information is stored in cleartext, attackers could potentially read it.

Even if the information is encoded in a way that is not human-readable, certain techniques could determine which encoding is being used, then decode the information.

Reproduction Steps

The following steps can be used for validation and remediation verification:

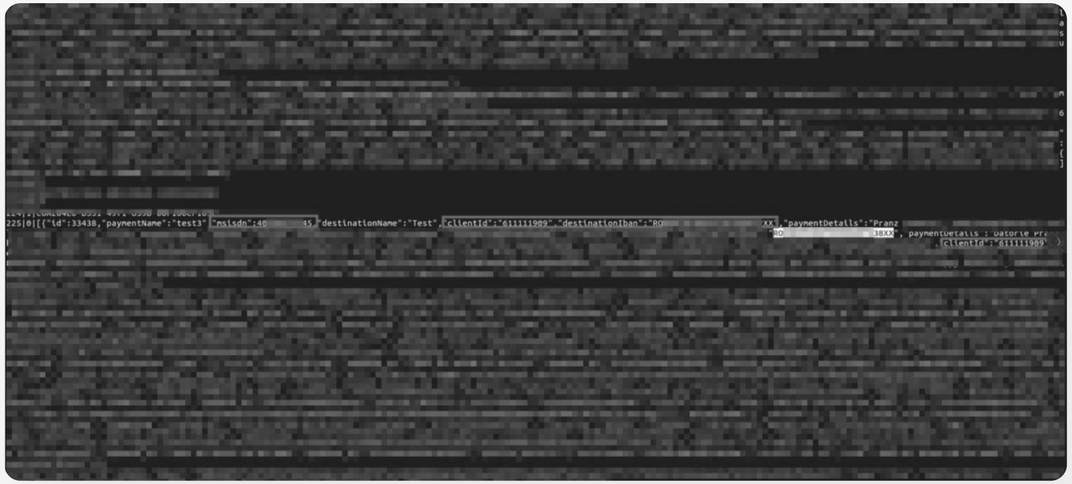
- Open the affected files (export the database)
- Search for sensitive information such as IBAN, access token, phone number, clientID

Impact

The severity of the error can range widely, depending on the context in which the product operates, the type of sensitive information that is revealed, and the benefits it may provide to an attacker. The business impact depends on the protection needs of the application and data.

Affected Entity	Sample · Identifier · SAM-0
Risk Statement	An attacker with access to the system could read sensitive information stored in cleartext
Affected Component	Mobile Identified Controls: None Identified
Residual Risk	Medium (CVSS Score: 5.3)
Classification	Cleartext Storage Likelihood: Medium
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	iOS Application

Vulnerability Details | V4



Sensitive information stored in Database

Recommendations

Moderate

- Avoid using plist and database files to store sensitive pieces of information as they can be easily retrieved by hackers.
- Data stored on the device should be considered “unsafe” and only a strong requirement can justify that storage.
- During the conception of a mobile app, one crucial thing is to clearly define what is stored in the application, why, and how.
- https://cwe.mitre.org/data/slices/919.html

Vulnerability Details

V5. Insecure data storage in Shared Preferences

Sensitive data is stored unencrypted in Shared Preferences files within the application private folder.

The SharedPreferences API is commonly used to permanently save small collections of key-value pairs. Data stored in a SharedPreferences object is written to a plain-text XML file.

The SharedPreferences object can be declared world-readable (accessible to all apps) or private. Misuse of the SharedPreferences API can often lead to exposure of sensitive data

Anyone with root-level access to the device will be able to see them, as the root has access to everything on the filesystem. Also, any application that runs with the same UID as the creating app would be able to access them (this is not usually done and you need to take specific action to make two apps run with the same UID, so this is probably not a big concern). Finally, if someone was able to mount your device's filesystem without using the installed Android OS, they could also bypass the permissions that restrict access.

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Open Share preferences file
- View content of the file for sensitive information

Impact

Insecure data storage can result in data loss, in the best case, for one user. In the worst case, for many users. From a business perspective, they can lead to Identity Theft, Fraud, Reputation Damage, External Policy Violation (PCI), or Material Loss.

Affected Entity	Sample · Identifier · SAM-1
Risk Statement	An attacker with access to the system might be able to read sensitive information
Affected Component	Mobile Identified Controls: None Identified
Residual Risk	Medium (CVSS Score: 5.1)
Classification	Cleartext Storage Likelihood: Medium
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	Android Application

Vulnerability Details | V5

Username and password stored in Shared Preference folder

Recommendations

Moderate

- Avoid using Shared Preferences and other mechanisms that can't protect data when you are storing sensitive information. Shared Preferences are insecure and unencrypted by default. You can use secure preferences to encrypt the values stored in Shared Preferences, but the Android KeyStore should be your first choice for storing data securely.
- If sensitive data needs to be stored inside the device then that can be stored encrypted, inside the database.
- https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d- Testing-Data-Storage.md

Vulnerability Details

V6. Application has no Jailbreak detection capabilities

It was discovered that the application allows jailbroken devices to install it, and use it correspondingly.

Jailbreaking iOS devices poses many security threats to the applications and the user. A jailbroken smartphone can easily be affected by malware and Trojans that can cause great damage to the application files. Hackers can easily install a tracking program on a jailbroken device to steal important files and information from the system. Apple recommends iPhone and iPad developers to detect if a device is jailbroken and deny the usages on such a device.

Reproduction Steps

The following steps can be used for validation and remediation verification:

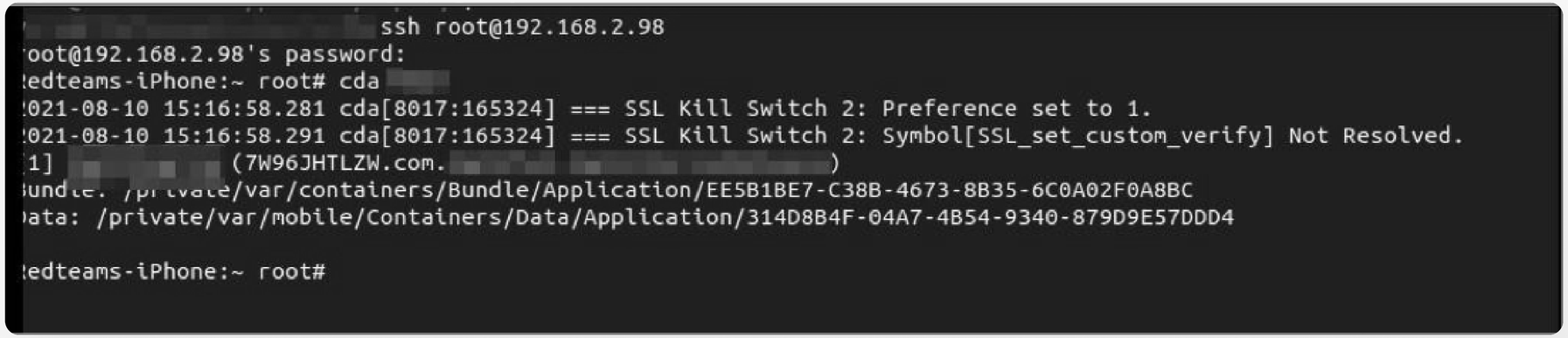
- Install the application on a jailbroken iOS device
- Start and browse the application

Impact

Allowing jailbroken devices to use the application will give to users much more information about the application itself and also, some of the security measures that are in place could be eliminated (e.g view traffic HTTPS)

Affected Entity	Sample Identifier SAM-3
Risk Statement	Installing the application on jailbroken device helps obtaining more information about the user or application and increases the attacking area.
Affected Component	Mobile Identified Controls: None Identified
Residual Risk	Medium (CVSS Score: 6.2)
Classification	Hardening Likelihood: Medium
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	iOS Application

Vulnerability Details | V6



The application folder can be accessed; no jailbreak detection activated

Recommendations

Complex

- We recommend not allowing Jailbreak devices to install and use the mobile application.
- Developers who stop users from running their apps on jailbroken devices could be a good idea from a security perspective, but this is really annoying to a techie user who is not able to execute the app only due to the reason that he has rooted his device.
- Jailbreaking techniques can be bypassed easily most of the time, and it is highly recommended that developers should use complex techniques in order to stop attackers from bypassing their validation controls.
- There exists only scattered information online about jailbreak detection methodology. This is partly because jailbreak detection is a sort of “special sauce.” Developers of mobile applications would rather keep their methodology private, and there are no real incentives to talking about it publicly.
- Most jailbreak detection methods fall into the following categories: File existence checks, URI scheme registration checks, Sandbox behavior checks, Dynamic linker inspection.
- https://duo.com/blog/jailbreak-detector-detector
- https://developer.apple.com/forums/thread/70603

Vulnerability Details

V7. Rooted device detection is easy to bypass

The security team detected that the application has root detection capabilities that do not allow it to be installed on rooted devices, but the detection mechanism is easy to bypass.

The security team is able to decompile the APK and alter the SMALI code in order to eliminate the function that detects if the device is rooted

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Decompile the APK
- Search in smali code all the root detection functions
- Change the functions in order to always return false.

Impact

The attacker can access the application folder and read sensitive information.

Affected Entity	Sample · Identifier · SAM-4
Risk Statement	An attacker might be able to bypass the detection mechanism
Affected Component	Mobile Identified Controls: None Identified
Residual Risk	Medium (CVSS Score: 4.3)
Classification	Hardening Likelihood: Medium
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	Android Application

Vulnerability Details | V7

Recommendations

Complex

- Use a more complex root detection functionality.

There are a few common ways to detect a rooted Android device:

- Check for test-keys (Check to see if build.prop includes the line ro.build.tags=test-keys indicating a developer build or unofficial ROM)
- Check for OTA certificates (Check to see if the file /etc/security/otacerts.zip exists)
- Check for several known rooted app's

  com.noshufou.android.su

  com.thirdparty.superuser

  eu.chainfire.supersu

  com.koushikdutta.superuser
- Check for SU binaries

  system/bin/su

  /system/xbin/su

  /sbin/su

  /system/su

  /system/bin/.ext/.su
- Attempt SU command directly (Attempt to run the command su and check the id of the current user, if it returns 0 then the su command has been successful)

- https://owasp.org/www-project-mobile-top-10/2014-risks/m10-lack-of- binary-protections
- https://medium.com/secarmalabs/comparison-of-different-android-root- detection-bypass-tools-8fd477251640
- https://medium.com/@cintainfinita/android-how-to-bypass-root-check-and-certificate-pinning-36f74842d3be

```
185
186     .line 46
187     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->detectTestKeys()Z
188
189     move-result v0
190
191     if-nez v0, :cond_1
192
193     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->checkSuExists()Z
194
195     move-result v0
196
197     if-nez v0, :cond_1
198
199     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->checkForRootNative()Z
200
201     move-result v0
202
203     if-nez v0, :cond_1
204
205     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->checkForMagiskBinary()Z
206
207     move-result v0
208
209     if-eqz v0, :cond_0
210
211     goto :goto_0
212
213     :cond_0
214     const/4 v0, 0x0
215
216     goto :goto_1
217
218     :cond_1
219     :goto_0
220     const/4 v0, 0x0
221
222     :goto_1
223     return v0
224 .end method
225
226 .method public isRootedWithBusyBoxCheck()Z
227     .locals 1
228
229     .line 66
230     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->detectRootManagementApps()Z
231
232     move-result v0
233
234     if-nez v0, :cond_1
235
236     invoke-virtual {p0}, [redacted]/rootbeer/RootBeer;->detectPotentiallyDangerousApps()Z
237
```

the isRooted() function will always return false

Changing root detection functions in smali code.

Vulnerability Details

V8. Application accepts any SSL Certificate (No SSL Pinning)

It was possible to intercept, view, and modify all traffic between the app and the server even though the traffic was sent over SSL.

A self-signed certificate was installed on the phone prior to testing. The app traffic was then redirected to a proxy server that presents the self-signed certificate.

The app, however, does not verify that the certificate is self-signed and still proceeds with allowing the user to access the service.

Reproduction Steps

The following steps can be used for validation and remediation verification:

- Add a Burp certificate on a jailbroken device and set the proxy in order for the traffic to go through Burp
- Explore the application
- All traffic can be seen in Proxy

Impact

An attacker would be able to intercept and edit the requests sent by the application to the server. This will increase the area of attack.

Affected Entity	Sample · Identifier · SAM-5
Risk Statement	Attackers will use this information to conduct more advance attacks and have access to sensitive data
Affected Component	Mobile Identified Controls: None Identified
Residual Risk	Medium (CVSS Score: 6.5)
Classification	Security Misconfiguration Likelihood: Medium
CVSSv3 code	https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Location	Android Application iOS Application

Vulnerability Details | V8

No SSL pinning protection activated; The traffic can be intercepted

Recommendations

Moderate

- Implement SSL Certificate Pinning which will stop the application if it detects a server certificate that is invalid or self-signed.
- https://owasp.org/www-project-cheat- sheets/cheatsheets/Pinning_Cheat_Sheet.html
- https://developer.android.com/training/articles/security-ssl
- https://developer.apple.com/news/?id=g9ejcf8y

Methodology

Security assessment involves looking for problems on the information systems being tested that may allow a malicious attacker to perform unwanted or undesirable actions. Information systems are comprised of a number of different software and hardware components. Errors in the configuration or programming of these components may create vulnerabilities, or potential weaknesses, that may allow an opportunity for an attacker to perform a malicious action. Different vulnerabilities require different levels of access or skill to be successfully used in a malicious way.

Andersen follows a highly structured methodology to ensure a thorough assessment of the system in scope and its environment is conducted. Our methodology uses a phased approach, consisting of information gathering, investigation, assessment, verification, and notification. Andersen employs a comprehensive and careful methodology in order to identify any potentially dangerous functionality. Prior to performing assessment against these functions, Andersen shares any potential impacts with the client. These steps ensure the least amount of business impact possible.

The Andersen Team will discuss a plan of attack as well as any potential concerns, and then will seek explicit approval from the client in order to proceed with the exploitation of any vulnerabilities that have the potential to impact production operations. The Andersen Team will communicate all verified vulnerabilities identified throughout the engagement that present significant danger to the client's organization. This will allow the client to begin planning remediation activities sooner, potentially closing the window on further exploitation by an attacker prior to the delivery of the final report.

Andersen follows industry best practice standards and methodologies when performing security-assessment activities. Such methodologies include:

- OpenSourceSecurityTestingMethodologyManual(OSSTMM)
- PenetrationTestingExecutionStandard(PTES)
- OpenWebApplicationSecurityProject(OWASP)TestingGuide
- TheNationalInstituteofStandardsandTechnology(NIST)
- PCIDataSecurityStandardPenetrationTestingGuidance(PCIDSS)
- TheIntelligenceLifecycle&F3EADCycle(ThreatIntelligence)
- OWASPMobileSecurityTestingGuide(MSTG)
- PenetrationTestingFrameworkforIoT(PTFIoT)
- PCIDSSATMSecurityGuidelines
- CISCloudFoundationsBenchmarkStandard
- OWASPCodeReviewGuide
- ThreatIntelligenceBasedEthicalRedTeamingFramework(TIBER-EU)
- ApplicationSecurityandDevelopmentSecurityTechnicalImplementationGuide
- SocialEngineeringAttackFrameworkandToolkit(SET)
- DigitalForensicsFramework(DFF)
- IncidentResponseFramework(NIST)
- SecureControlsFramework(SCF)
- CRESTPenetrationTestingGuide
- CSASTARSelf-Assessment/CAIQ
- CISSecurePlatformsBenchmarks(CISSecurity)
- ApplicationSecurityVerificationStandard(ASVS)

Application Security Assessment Methodology

An Application Security Assessment (ASA) is an assessment of the application on the running device as well as the associated back end components. The objective of the assessment is to identify vulnerabilities within these environments and verify their existence using manual testing techniques. These assessments are most successful when customers share all known information with the tester; however, the customer can elect to share less information.

The MASA is a comprehensive assessment that identifies vulnerabilities ranging from Critical to Informational severity. Andersen's Application Security Team identifies, verifies, and reports anything that raises the attack surface of the application. We will use multiple techniques to simulate attacks from both an unauthenticated and authenticated attacker perspective, exposing the greatest amount of attack surface and providing the most value from the testing efforts.

After the client provides the initial information and the application, the Andersen Team will run the application and test the credentials taking note of their privilege levels. During the initial review of the application, the consultant will take note of any critical areas of the application and identify potentially sensitive functions to target during the testing efforts.

The Andersen consultant will run and analyze the mobile application on a jailbroken or rooted device, provided by Andersen and connected to a customized assessment environment compromised of wireless access points, proxies, and many other tools. If the consultant does not use a Andersen- provided device, the client must supply a suitable test device. Using a jailbroken or rooted device simulates a potential attacker's approach for attacking an application or accessing sensitive data on users' mobile devices. The Andersen team uses a methodology that focuses on real-world scenarios to provide an accurate risk assessment.

In some instances, certain mobile security features may limit the application functionality on a testing device. These may include jailbreak or root detection, tamper prevention, device provisioning, or certificate pinning. When such features are present, we recommend providing an alternate build of the application with these features disabled or removed. The Andersen consultant will validate the strength of the security controls in the original application, but if the consultant cannot defeat these in a reasonable period of time, the consultant will continue testing with the alternate version. This approach allows for in-depth testing of the application within the time allotted.

Application Security Assessment Methodology

The methodology employed during a Mobile Application Security Assessment involves the following stages

Workflow Study

Map the Content
- Explore Visible Content and Discover Hidden Content
- Discover Default Content
- Identify Specific Function

Analyse
- Identify Functionality
- Identify Data Entry Points
- Identify Technologies

Vulnerability Identification

Test Auth & Access Controls
- Username & Password attacks
- Testing with limited access, multiple accounts
- Test for insecure methods

Test Session Management
- Session Fixation and Termination
- XSRF
- Mapping of token to sessions

Test Client Side Controls
- Hidden Files, Cookies
- Pre-set Parameters, Length Limits
- Tick Client Components

Test Input Validation
- SQL Injection, Cross Site Scripting
- Path Traversal, Script Injection
- File Injection

Test Logic Flaws
- Incomplete Input
- Trust Boundaries
- Transaction Logic

Test Server Flaws
- Configuration Checks
- Payload Checks

Exploitation

Tasks performed
- Identify Business Risks
- Building Test Risk
- Exploitation and verification

Reporting

Tasks performed
- Executive Summary
- Technical Summary
- Clarification and Results Presentation

Application Security Assessment Methodology

The steps are aligned to the in-depth security concepts and are focused on process and technical security controls and their implementation in the various phases of the project delivery. The results provided for each activity will include a detailed and comprehensive assessment of client's current security posture, expansive recommendations, and tools and knowledge to facilitate the continuous improvement.

Conduct passive and active information gathering to determine the level of information that can be found about the assets in scope. These actions are conducted in order to understand what level of exposure the assets have, and how an attacker can use this information to conduct further attacks.

Intelligence Gathering & Workflow Study

Vulnerability Assessment & Identification

Security Engineers investigate for vulnerabilities through manual searches complemented by automated tools. The objective is to discover as many vulnerabilities as possible on the target.

Exploitation

The exploitation phase consists in testing possible exploitations of the flaws identified in the previous phase. This step allows using certain flaws as "pivots", in order to discover new vulnerabilities. The exploitation of security vulnerabilities allows evaluating their real impact and thus their criticality level.

Reporting

The Report will communicate to the reader the specific goals of the Penetration Test and technical details of findings of the assessment exercise. The intended audience will be those who are in charge of the oversight and strategic vision of the security program as well as any members of the organization, which may be impacted by the identified/confirmed threats.

Testing

Andersen conducts testing using both automated and manual testing methods. A majority of the testing will focus on manual testing. Andersen uses automated testing, where possible, on back end and web service components, and manually validates identified vulnerabilities. Throughout the assessment, Andersen uses a mixture of commercial, open source, and custom-built tools.

Application Security Assessment Methodology

The Andersen Team configures the automated tools using information obtained during the information-gathering phase. This ensures the highest level of success, by removing obstacles that typically hinder their functionality and ability to navigate the application properly.

Andersen security checklist for Mobile Applications includes, but is not limited to, identification of the following risks:

OWASP TOP 10 MOBILE RISKS
M1: Improper Platform Usage
M2: Insecure Data Storage
M3: Insecure Communication
M4: Insecure Authentication
M5: Insufficient Cryptography
M6: Insecure Authorization
M7: Client Code Quality
M8: Code Tampering
M9: Reverse Engineering
M10: Extraneous Functionality

Examples of tools that Andersen uses:

Burp Suite Pro · Apktool · dex2jar · Android Studio · JD-Gui · Frida · Hopper App · Xcode · iExplorer · SQLite Studio · adb · ssh · Cydia · Pangu · TaiG · cycript · Wireshark

Application Security Assessment Methodology

Andersen performs extensive manual testing, which comprises a significant majority of the testing effort. During this portion of the testing, the Andersen consultant executes the application, and analyzes the communication, functions, and the data the application sends and receives. The Andersen Team tests complex interactions, workflows, and business logic. Additionally, the Team manually evaluates areas of the application and specific vulnerabilities that automated tools either have difficulty with or are unable to identify.

APP PROFILING AND INFORMATION DISCLOSURE
Default Banners
Unhandled Error Conditions
HTML/JavaScript Comment Information Leakage
Extraneous Contentin Web Root
Source Code Disclosure
Robots.txt Path Disclosure
Content Expiration and Cache Control
Account Enumeration
Backup/Archive Content
Bit Bug/Referrer Header Leakage

PLATFORM AND THIRD-PARTY MISCONFIGURATION
Default Administrative Credentials
Default Contentand Scripts
Application Script Engine
Web Server
Weak SSL Implementation
Flawed Use of Cryptography

COOKIE AND SESSION HANDLING
Session Fixation/Hijacking
Set-Cookie Weaknesses
Sensitive Information Disclosure
Cookie Poisoning
Multiple Simultaneous Login Allowed
Session Timeout
Explicit/Implicit Logout Failures
Cookieless Sessions
Custom Session Management

Application Security Assessment Methodology

Andersen validates any identified communication channels for proper confidentiality and integrity. We monitor the application execution on the device, and examine the device from a high-level forensic perspective to identify areas where the application may be storing or caching sensitive information in an insecure manner. After reverse engineering the application binary to the extent possible, we analyze it for information leakage or hard-coded secrets. During testing, we map the back-end environment and test any in-scope components for vulnerabilities.

COMMAND INJECTION FLAWS
SQL Injection
XXE, XPath, and XML Injection
SSI/OS Command Injection
Server Script Injection/Upload
Cross-Site Scripting (XSS)
Buffer Overflow

LOGIC FLAWS
Privilege Escalation
Sensitive Information Disclosure
Data Mining/Inference
Functional Bugs
Application-Specific Control Failures
Cross-SiteTracing (XST)
Weak Data Validation
Race Conditions
CPU-Intensive Functions

CLIENT-SIDE FLAWS
Exposure of Sensitive Business
Logic
Relianceon Client-Side Validation
AJAX/Web Service Flaws
Java Applet/ActiveX
Control/Flash Weaknesses

AUTHENTIC. AND AUTHORIZATION
Unauthenticated Sensitive Content
Poor Separation of Privilege
Brute-Force Login
Weak Password Policy
Account Lockout/Denial of Service
SSO Weaknesses
Security Question Weaknesses
CAPTCHA Flaws

Threat Classification and Reporting

When any exploitable vulnerability is discovered, further research is conducted on that vulnerability to identify its level of severity. The risk is calculated according to the following criteria:

Impact

The security impact on the web application in the event of an exploitation of this vulnerability by an attacker. This criterion indicates the benefit of the attack to the attacker.

Popularity and Ease of Identification of the Vulnerability

This criterion factors in the public availability of information and tools to detect the vulnerability. Problems that have easy to use exploit code available on the Internet, for example, would get a higher rating. This criterion indicates the probability of an attack.

EaseofExploitation

The level of difficulty for an attacker to exploit this problem.Difficulty could increase due to technical complexity, the need for prior knowledge of the network, or other factors. This criterion indicates the cost in time and resources of the attack for the attacker.

After identification and classification of the findings is complete, the details of each finding will be documented and detailed recommendations will be given on how to mitigate the discovered threats.

Threat Classification and Reporting

The risk is classified as follows:

Risk Classification	Description
Critical	Vulnerabilities in this category usually have the following characteristics: • Exploitation of the vulnerability results in root/administrator-level access to the system; • The information required in order to exploit the vulnerability, such as example code, is widely available to attackers; • Exploitation is usually straightforward, in the sense that the attacker does not need any special authentication credentials or knowledge about individual victim systems, and does not need to persuade a target user, for example via social engineering, into performing any special functions.
High	Vulnerabilities that score in the high range usually have the following characteristics: • The vulnerability is difficult to exploit; • Exploitation does not result in elevated privileges, but may grant unintended access to data; • Exploitation does not result in a significant data loss.
Medium	Vulnerabilities that score in the medium range usually have the following characteristics: • Denial of service vulnerabilities that are difficult to set up; • Exploits that require an attacker to reside on the same local network as the victim; • Vulnerabilities that affect only nonstandard configurations or obscure applications; • Vulnerabilities that require the attacker to manipulate individual victims via social engineering tactics; • Vulnerabilities where exploitation provides only very limited access.
Low	Vulnerabilities in the low range typically have very little impact on an organization's business. Exploitation of such vulnerabilities usually requires local or physical system access.
Informational	These are not vulnerabilities, but additional information gleaned from the target during vulnerability testing.

Risk Calculation

Andersen utilize the Basic Common Vulnerability Scoring system (“CVSS”) version 3 by default for Residual Risk calculation, which takes into consideration the following criteria:

Criteria	Description
Attack Vector	This metric indicates how ‘close’ an attacker needs to be to the object. Is physical access needed at one end (AV:P)? Or can the object at the other end be attacked via the network?
Attack Complexit	How easily can the attacker reach their target? Is it with in their control?
Required Privileges	Does the attacker need privileges (authorization) before they can carry out their attack? If this is the case, the score is lower, otherwise, it is higher.
User Interaction	Must a user do anything first before the attacker reaches their target? If the user, for example, has to click on a link first, the value would be ‘required’ (UI:R).
Scope	The scope describes whether the effects of an attack ‘only’ affect the vulnerable components or other components. In the last case (‘changed’ S:C), the scope score increases the base score if the latter has not already reached the maximum value of 10.
Confidentiality Impact	This metric indicates to what extent the attack affects confidentiality. A ‘high’ (C:H) value means that confidentiality has been totally lost.
Integrity Impact	In the same way, this metric describes the influence on the integrity of the data. If, for example, the attackers were able to modify all files, the impact would be set to ‘high’ (I:H).
Availability Impact	This measure is also very similar to the other impact metrics.If the attacker succeeds or were able to succeed in denying the availability of the components so that they can no longer be accessed, the maximum value ‘high’ (A:H) would be reached.

Thank you for your attention!

Bringing automation and clarity to Cyber Security

andersenlab.com