Sample Company Ltd.

Report for Sample Application Mobile Application Pentest

January 25, 2024

Document Details

Table of Contents:

The following gives detailed information about the structure of the document:

Document History:

The following gives an overview of the release history of this document:

Table 1 - Document History

Release Date	Description	Author
January 24, 2024	Initial Release	Naunet John
January 25, 2024	Revision 1	Naunet Jane

Distribution List:

The below table contains the list of client employees who are the designated recipients of this document:

Table 2 – Distribution List

Name	E-mail Address
John Doe	john.doe@client.com
Jane Doe	jane.doe@client.com

Restrictions and Legal

This document and the information contained within is confidential and provided solely for Sample Company Ltd's informational purposes and internal use.

This document is not intended to be and should not be used by any person or entity other than Sample Company Ltd. without the explicit written permission of Naunet.

Sample Company Ltd. may choose to share the information contained in this report with auditors of relevant compliance frameworks under non-disclosure agreement.

Any unauthorized copying, distribution, and publishing, or dissemination of the information including sharing with any third party is strictly prohibited.

These materials and the information contained herein are provided by Naunet and are intended to provide general information on a particular subject or subjects and are not an exhaustive treatment of such subjects.

Accordingly, the information in these materials is not intended to constitute any professional advice or services. The information is not intended to be relied upon as the sole basis for any decision which may affect Sample Company Ltd's business.

These materials and the information contained therein are provided as is, and Naunet makes no express or implied representations or warranties regarding these materials, or the information contained therein. Without limiting the foregoing, Naunet does not warrant that the materials or information contained therein will be error-free or will meet any criteria of performance or quality. Naunet expressly disclaims all implied warranties, including, without limitation, warranties of merchantability, title, fitness for a particular purpose, non-infringement, compatibility, security, and accuracy.

Sample Company Ltd's use of these materials and information contained therein is at your own risk, and you assume full responsibility and risk of loss resulting from the use thereof. Naunet will not be liable for any special, indirect, incidental, consequential, or punitive damages or any other damages whatsoever, whether in an action of contract, statute, tort (including, without limitation, negligence), or otherwise, relating to the use of these materials or the information contained therein.

Differently form the above written in case the information and materials are expressly provided as final performance of a contract concluded between Sample Company Ltd and Naunet, Naunet takes liability that the service has been provided and the product - if any - has been prepared contractually. Naunet excludes all liability for damages arising out of or in connection with the documents, materials, information, and data provided by Sample Company Ltd. For all the questions not ruled herein, the relating contract shall be applicable.

In this document Sample Company Ltd. may also be referred to as "Sample Company" or the "company" or in context as the "client", as well as VoidSec Kft. may also be referred to as "Naunet".

If any of the foregoing is not fully enforceable for any reason, the remainder shall nonetheless continue to apply completely.

VoidSec Kft. Contact Information:
info@naunet.eu
Hungary 2903
Komárom
Tamási Áron utca 4.

Assignment Summary

Assignment Summary

Sample Company Ltd. engaged VoidSec Kft. for its services to perform a mobile application information security penetration test of the “Sample Application”.

The primary objective of the assessment was to identify vulnerabilities or configuration weaknesses, which could lead to unauthorized access or data theft, with a focus on finding weaknesses that are available to anyone with network-level access to the application.

Following the identification of vulnerabilities during this engagement, Sample Company is advised to implement appropriate measures to remediate these vulnerabilities and enhance the overall security posture of their application and the associated network infrastructure.

Assignment Disclaimer

We performed our testing from January 2, 2024, to January 6, 2024, and afterwards, we created this report and finalized the supporting documentation. Consequently, the findings presented in this report reflect the security status of the tested target systems and applications as of the last day of the testing period and should be treated as a snapshot in time.

Management Summary

The scope of the assessment was defined as the following:

- **Mobile Application Penetration Test** for the UAT (User Acceptance Test) version of **Sample Application** iOS and Android applications.

During the Mobile Application Penetration Test two high-risk issues were identified.

Lack of Certificate Validation: We found that the application used encrypted HTTPS communication to transmit data to the server. However, we observed that the application accepted non-trusted (e.g., self-signed) certificates. This might allow an attacker to perform Man-in-the-Middle (MitM) attack, to capture sensitive information between the client and the server.

Recommendation: We recommend enforcing a strict SSL certificate policy, where untrusted certificates are not accepted, and users are not allowed to accept malicious certificates or manually override the policy.

Assignment Conclusion

We recommend mitigating the misconfigurations described in this report and performing a re-test to validate whether applied fixes are effective.

The detailed description of all found issues, the associated risks, and the recommendations to reduce the risks can be found in the following parts of this document.

Assignment Details

Assignment Background

Naunet was tasked to conduct a security assessment, which included a mobile application penetration test of UAT version of the Sample Application mobile application. Confidentiality, integrity, and availability of IT assets play a vital role in the business model and processes of Sample Company.

Assignment Goal

The project goal was to identify any existing security vulnerabilities, which might be exploitable by an attacker with network-level access such as a malicious employee.

Scope of the Mobile Application Penetration Test

The following gives an overview specified as the scope of the Mobile Application Penetration Test.

The following application have been designated as the scope:

Table 3 - Scope of the Mobile Application Penetration Test

Name and version	Environment
Sample Application 1.0 for Android	UAT
Sample Application 1.0 for iOS	UAT

The tested application used the UAT environment available on the intranet with authentication required. The functionality of the application is to allow users to manage scenarios for financial reporting.

Two test user accounts were registered for the purposes of the test, the following table contains the account details:

Table 4 - Used test accounts

Username	User Role
Username1	Administrator
Username2	Low-Privilege Role

Performed Services

The following gives an overview of the performed services:

- **Mobile Application Penetration Test**, where the assignment was addressing the “Sample Application” mobile application for iOS and Android. The scan included a vulnerability assessment of the mobile application with manual and semi-manual methods in a grey box approach.

Methodology

While conducting the assignment we applied our testing methodology, aligned with the NIST recommendations outlined for penetration testing (NIST Special Publication 800-115, NIST Special Publication 800-12 Revision 1), as well as processes outlined by various publications made by ENISA.

We have performed all typical tests (where applicable) outlined in the OWASP Mobile Application Security Testing Guide (Version 1.7.0), including but not limited to the following:

- Application Architecture Testing
- Data Storage and Privacy Testing
- Testing for Weak Cryptography
- Authentication Testing
- Session Management Testing
- Network Communication Testing
- Client-Side Testing
- Input and Data Validation Testing
- Tampering Resistance Testing
- Reverse Engineering Resistance Testing
- Verification of Code Quality and Build Settings

During the assignment no attempt was made to perform the necessary tests in a covert manner.

The performed testing does not necessitate any cleanup on the target systems in scope of the assignment.

For more information on the limitations and reasoning behind not performed test types please see the following section.

The list of applied tools for the analysis can be found in the Testing Infrastructure and Tool Details section of this document.

Limitations Faced During the Test

The following list gives an overview of the limitations faced during the project:

- The scope of the penetration test did not include the following functions of the application: “Function1”, “Function2”, “Function3”, ”Function4” and “Function5”, the aforementioned functionalities were out of scope.

Findings of the Mobile Application Penetration Test:

The following table gives an overview of our findings of the Mobile Application Penetration Test.

Table 5 - Findings of the Mobile Application Penetration Test

Finding ID	Name	Issue Type	Vulnerability Type	Risk	Status
AND-1	Authentication Bypass	Application	Insecure Authentication/Authorization	Medium	Open
AND-2	Lack of Certificate Pinning	Application	Insecure Communication	Low	Open
AND-3	Lack of Hook Detection	Application	Insufficient Binary Protections	Low	Open
IOS-1	Lack of Certificate Validation	Application	Insecure Communication	High	Open
IOS-2	Insecure Keychain Settings	Application	Security Misconfiguration	Medium	Open
IOS-3	Insecure Biometric Authentication Implementation	Application	Insecure Authentication/Authorization	Low	Open

Detailed Findings

The following section contains the detailed technical descriptions of the misconfiguration or security weakness related findings of the assignment.

AND-1: Authentication Bypass

Targets	Finding Status	Overall Risk	Medium
Sample application - Android 1.0	Open	Impact	High
		Likelihood	Low

Description

Sample Company's Sample application was designed to store documents in a secure way. Users could set password authentication in the application to decrypt and encrypt their documents on the company devices. We found that the authentication module was insecurely implemented and could be bypassed by modifying the "lib/libvault.so" file.

The authentication module checked for the user's password and if the password was incorrect, the application did not decrypt the documents. However, it was possible to bypass the branch in the application, where the application handled the incorrect password. This way the application decrypted the user's documents, even if an invalid password was provided.

To reproduce the issue, modify the BLT instruction at the 000cdb54 address to a NOP instruction preventing the application from reaching the invalid credentials branch. The following screenshot shows the original instructions:

```
                    LAB_000cdb40
000cdb40 01 0a 20 ee     vmul.f32    s0,s0,s2
000cdb44 c0 0a bd ee     vcvt.s32...  s0,s0
000cdb48 84 02 9a e5     ldr         r0,[r10,#0x284]
000cdb4c 88 12 9a e5     ldr         r1,[r10,#0x288]
000cdb50 01 00 50 e1     cmp         r0,r1
000cdb54 1d 00 00 ba     blt         LAB_000cdbd0
000cdb58 3e 00 00 ea     b           LAB_000cdc58
000cdb5c 5c 5d 3c 00     undefined4  003C5D5Ch
000cdb60 a4 5d 3c 00     undefined4  003C5DA4h
```

Figure 1 – Original instructions

The following screenshot shows the modified instructions:

```
                    LAB_000cdb40
000cdb40 01 0a 20 ee     vmul.f32    s0,s0,s2
000cdb44 c0 0a bd ee     vcvt.s32...  s0,s0
000cdb48 84 02 9a e5     ldr         r0,[r10,#0x284]
000cdb4c 88 12 9a e5     ldr         r1,[r10,#0x288]
000cdb50 01 00 50 e1     cmp         r0,r1
000cdb54 00 f0 20 e3     nop
000cdb58 3e 00 00 ea     b           LAB_000cdc58
000cdb5c 5c 5d 3c 00     undefined4  003C5D5Ch
000cdb60 a4 5d 3c 00     undefined4  003C5DA4h
```

Figure 2 – Modified instructions to skip password validation

Impact

An attacker might be able to access sensitive information by decrypting the stored documents.

Likelihood

An attacker needs access root access on a company device to modify the application. The likelihood was lowered by the fact that the company devices were enrolled in Sample Company's Mobile Device Management (MDM) solution.

Recommendation

We recommend redesigning the application using obfuscation and implementing a more complex authentication solution.

References

N/A

AND-2: Lack of Certificate Pinning

Targets	Finding Status	Overall Risk	Low
Sample application - Android 1.0	Open	Impact	Low
		Likelihood	Low

Description

We found that the application used encrypted HTTPS communication to transmit data to the server. However, we observed that after adding the certificate authority (CA) certificate of our intercepting proxy to the Android trust store of the device, the application connected to the backend server and we were able to eavesdrop on the communication between the application and the server.

Impact

If an attacker is able to install a rogue certificate authority to the "system" trust store of the device, he is able to eavesdrop on the HTTPS traffic between the application and the server. The attacker could steal the session token and replay or forge new requests sent to the server.

Likelihood

The attacker would need to perform the following steps to abuse this vulnerability:

1. Using OpenSSL, an attacker generates a new certificate authority (CA) certificate.
2. The attacker sends a forged e-mail to the user. The mail is masqueraded to look like it came from the IT support personnel and entices the user to add the attached CA file to the Android credentials trust store.
3. The user falls for the scam mail and installs the certificate. As a result, a new CA certificate is added to the device operating system level CA store.
4. Using ARP poisoning technique, the attacker could intercept the traffic on the same network, or the victim user connects to a rouge network controlled by the attacker.
5. The attacker performs a Man-in-the-Middle attack using the installed rouge certificate.

After the steps above, the attacker is able to sniff the traffic of the application.

We note that above Android 7.0 (Nougat) the installation of system level CA certificates requires root access on the device.

Recommendation

Instead of relying only on the Android CA trust store, we recommend enforcing the check of specific CA fingerprints on the communication SSL endpoints (SSL certificate pinning). We also recommend enforcing a strict SSL certificate policy, where users are not allowed to accept malicious certificates or manually override the policy. Ideally, a check for the presence of a particular certificate authority within the certificate trust chain (known as certificate pinning) is recommended.

To implement certificate pinning and allow for rotation of server certificates without releasing new application versions, pin the modulus of the server certificate’s public key, that represents a unique identifier for the private key. Comparing the public key modulus and the private key modulus, the mobile application trusts the eligible certificate (avoid trusting any other certificates). Using the modulus allows to change the server certificate as long as the private key does not change.

Implementing a certificate pinning capability just “raises the bar” for an attacker and cannot fully mitigate the associated risk.

References

N/A

AND-3: Lack of Hook Detection

Targets	Finding Status	Overall Risk	Low
Sample application - Android 1.0	Open	Impact	Low
		Likelihood	Low

Description

We found that the Android application did not utilize protection against hooking. As a result, we were able to circumvent several security features in the application and read sensitive data from the running application. We were able to bypass root detection, allow screenshots to be made and toggle the application's debug mode. For bypassing root detection we used the following code with Frida:

```
Java.perform(function () {
    var RootDetection = Java.use("Security");
    RootDetection.RootChk.overload().implementation = function(){
        console.log("[*] Root detection bypassed!");
        return ``"~!@#$%^&*(){}[]" or false or any malformed request``;
    };
});
```

Impact

Without hooking protection, reverse engineers can step through the code, stop the execution and inspect the state of variables, read and modify the device's memory. It also allows the attackers to bypass security implementations, such as root detection.

Likelihood

An attacker with access to rooted device with the installed application can conduct the attack.

Recommendation

We recommend implementing hook detection mechanisms in the application, like checking the installed hooking framework applications on the device using PackageManager. Also check the presence of the hooking frameworks in stack trace method calls by looking after "frida", "xposed.XposedBridge" and "saurik.substrate" strings.

Please be aware that implementing hook detection just "raises the bar" for an attacker and cannot fully mitigate the associated risk.

References

N/A

IOS-1: Lack of Certificate Validation

Targets	Finding Status	Overall Risk	High
Sample application - iOS 1.0	Open	Impact	High
		Likelihood	Medium

Description

We found that the application used encrypted HTTPS communication to transmit data to the server. However, we observed that the application accepted non-trusted (e.g., self-signed) certificates.

The following function call caused the issue:

```
+[NSURLRequest allowsAnyHTTPSCertificateForHost:@"mobile.samplecompany.com"]
```

Impact

The attack could end up in unauthorized disclosure of sensitive data or the attacker may alter the commands sent to the webserver and/or the web server responses.

Likelihood

An attacker may be able to perform the man-in-the-middle attack against the application data flow.

Recommendation

We recommend enforcing a strict SSL certificate policy, where untrusted certificates are not accepted, and users are not allowed to accept malicious certificates or manually override the policy. Moreover, checking for the presence of a particular certificate authority within the certificate trust chain (known as certificate pinning) is also recommended.

References

N/A

IOS-2: Insecure Keychain Settings

Targets	Finding Status	Overall Risk	Medium
Sample application - iOS 1.0	Open	Impact	High
		Likelihood	Low

Description

We were able to read out the PIN code of the application and the authentication tokens, which were stored in the Keychain database.

During the test, we found the application set *“kSecAttrAccessibleAlways”* attribute on all keychain item. This insecure setting leads to authentication bypass and access to the compromised account. However to exploit this issue some prerequisite has to be fulfilled.

The following steps describe the prerequisites:

1. The victim turns the Passcode on and set a four or six digit number up
2. They open the `Sample Application` application and enables the PIN code under the “My account/Security and Password/” menu
3. The victim turns the Passcode off

Then an attacker needs to perform the following actions to exploit the issue:

1. The attacker needs to have physical access to the jailbroken device
2. Then they are able to read out the PIN code of the application and the authentication tokens. For the Proof of Concept we read out the PIN code with Passionfruit.
3. After the attacker reads out the PIN code, they need to turn the Passcode on and enter the PIN code when the authentication window appears in the application.
4. Finally the attacker bypass the authentication and has full control over the victim account

We note that, the application did not allow us to authenticate with PIN code and biometric methods when the device’s Passcode was turned off. Therefore, the attacker needs to re-enable it to bypass the authentication.

Impact

If an attacker gets physical access to a device, they are able to login to the account of the victim user and compromise it.

Likelihood

An attacker needs physical access to a jailbroken device and the device Passcode needs to be turned off.

Recommendation

We recommend setting the accessible attribute to “kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly” on all keychain item.

References

More information about the keychain accessible attribute:
https://developer.apple.com/documentation/security/ksecattraccessiblewhenpasscodesetthisdeviceonly

IOS-3: Insecure Biometric Authentication Implementation

Targets	Finding Status	Overall Risk	Low
Sample application - iOS 1.0	Open	Impact	Low
		Likelihood	Low

Description

We found that the fingerprint authentication feature handled the Face ID insecurely in the application. By hooking to the -[LAContext evaluatePolicy:options:reply:] function we were able to bypass the Face ID authentication and login without providing the correct biometric face identity (Face ID) or the knowledge of the user's PIN code.

We used the following proof of concept Frida code to bypass the Face ID on an iPhone `X` device (IOS `11.3`, jailbroken):

```
if (ObjC.available) {

      var hook = ObjC.classes.LAContext["- evaluatePolicy:localizedReason:reply:"];

      Interceptor.attach(hook.implementation, {

              onEnter: function(args) {
                      console.log("[+]: hooking Touch ID");

                      var block = new ObjC.Block(args[4]);
                      var callback = block.implementation;
                      block.implementation = function(error, value) {
                              var reply = callback(1, null);
                              return reply;
                      };
              }
      });

} else {
      send("[-]: Objective-C Runtime is not available!");
}
```

To bypass the Face ID authentication we tried to authenticate with the front camera of the device. After unsuccessful attempts we tapped on the "Cancel" button on the dialog windo.

After we tapped on the Cancel button, the application called the -[LAContext evaluatePolicy:options:reply:] method which was overridden in our Frida hook, therefore we logged in without verification.

Impact

An attacker with physical access to the unlocked jailbroken device could login in the name of the user without providing a correct fingerprint or the password for the application.

Likelihood

The attacker needs physical access to a jailbroken device with unlocked screen and a pre-configured application.

Recommendation

We recommend utilizing the KeychainServices API for securely storing the token. We also recommend setting the "user presence" (kSecAccessControlUserPresence) policy for the keychain entry and the protection class of kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, so that the fingerprint token will be encrypted until the user provides the correct Face ID or passcode.

References

More information on using the Keychain for Touch ID integration can be found at the following URL:
https://www.synopsys.com/blogs/software-security/integrating-touch-id-into-ios-applications/

Testing Infrastructure and Tool Details

Description of the testing infrastructure:

The below table contains the details of the operating systems used during the assignment:

Table 6 - Details of the operating systems used during the assignment

Operating System	Used Version
Kali Linux	Rolling 2023.1
MacOS Ventura	Version 13.1

We performed our external facing test using the following public IP addresses:

Table 7 - Details of the public IP address used during testing

IPv4 Address	ISP	ASN	Country	City
81.183.236.252	Magyar Telekom	AS5483	Hungary	Budapest

List of Tools used during testing:

The below table gives an overview of the specific tools we have used during this specific assignment.

Note: Generic operating system utilities such GNU packages within a Linux distribution will not be listed below, furthermore if a tool has been updated or upgraded during the testing period, only the latest used version information will be listed below.

Table 8 - Details of the tools used during the assessment

Tools Name	Description	Used Version
Tool #1	Tool Description #1	Version 1.0.
Tool #2	Tool Description #2	N/A

Finding Risk Details

Risk Details

We use the following risk ratings: Informational, Low, Medium, High and Critical. These categories offer limited accuracy however they represent that during an assignment the complete context and relevant technical information related to findings may only be partially available.

During the risk calculation process due to the limited information, time and resources of an assignment and specific goal, we do not “compress” or “inflate” the calculated findings to be accurate on an organizational level. The calculated risk does not represent a potential priority for mitigation on an organizational level, as other risks may be present in out-of-scope systems.

The Informational category is reserved for findings that are by themselves do not represent direct risk to Sample Company, however the finding’s existence may hold valuable information.

Given that new vulnerabilities in operating systems and applications are discovered daily and system configurations change frequently, these results are based on publicly available vulnerability information and the system configurations at the time of testing. Consequently, new vulnerabilities may have emerged since then, or existing weaknesses may have been addressed.

Customers should perform their own assessments to align these findings with their information security management risk rating methodology. Additionally, customers should determine if and how to implement the recommended remediation activities. A thorough risk rating requires internal knowledge of the company's business requirements and organizational processes.

Note: Please note that the report may include "False Positives" due to a lack of access to all necessary system or design information. Therefore, all results must be verified to confirm their corresponding risk.

The following Risk Calculation Matrix describes how the risk rating and potential advice for further action is derived from the Likelihood and Impact:

Impact \ Likelihood				
High	Medium	High	Critical
Medium	Low	Medium	High
Low	Low	Low	Medium
Informational	Low	Medium	High

Table 9 – Risk Calculation Matrix

Impact Details

Quantifying potential damage without a detailed financial risk assessment is challenging, so we use a scenario-based approach. To provide a comprehensive overview of the risk environment in the audited areas, we consider various factors, including but not limited to the confidentiality, integrity, and availability of information; the impact on technical operations; the potential effects on reputation and customer satisfaction; the risk of potential penalties or legal consequences; the financial impact; the implications for customer safety; and other relevant aspects.

Our professional judgement, based on our experience, leads us to categorize vulnerabilities by their potential impact.

High: Findings that enable attackers to manipulate sensitive information, damage Sample Company's reputation, obtain unauthorized access to sensitive information (such as financial information or Personal Identifiable Information), or obtain unauthorized access to the applications or infrastructure of Sample Company are to be categorized as high impact.

Medium: Medium impact findings are that may still allow attackers to damage Sample Company's reputation or obtain unauthorized access to Sample Company's information, however the privileges an attacker could obtain by taking advantage of these vulnerabilities are limited and are likely to depend on other weaknesses and other resources to be usable for accessing sensitive information.

Low: Low impact findings do not pose an immediate threat, but may ultimately expose sensitive information, offering valuable intelligence to potential attackers. These vulnerabilities provide only restricted access to the system and limited operational risk to Sample Company.

Note: Please note that in the described system, impact by itself cannot be classified as critical.

Likelihood Details

The likelihood of a vulnerability being exploited is assessed based on three critical factors: the ease of discovery, the number of potential attackers with access to it, and the resources and expertise needed to execute an exploit. The probability of exploitation is then categorized as follows:

High: Findings with high likelihood can be easily identified and exploited by malicious actors. They are broadly accessible to numerous attackers, requiring only minimal technical expertise and resources to leverage.

Medium: Medium-likelihood findings require a more advanced approach for exploitation. They are not easily detectable and are accessible to a limited pool of attackers. Exploiting these vulnerabilities demands higher technical expertise or significant resources.

Low: Low-likelihood vulnerabilities are challenging to exploit due to their limited discoverability, restricted access to a small number of potential attackers, or the need for advanced expertise and sophisticated resources to execute an exploit.

Note: Please note that in the described system, likelihood by itself cannot be classified as critical.