https://e-discovery.ng/

ANDROID APPLICATION PENETRATION TESTING

Report for:	
Date:	

This document contains confidential information about IT systems and network infrastructure of the client, as well as information about potential vulnerabilities and methods of their exploitation. This confidential information is for internal use by the client only and shall not be disclosed to third parties.

Table of Contents

Executive Summary

E-Discovery (Provider) was contracted by _____ (Client) to carry out an **android application** penetration test.

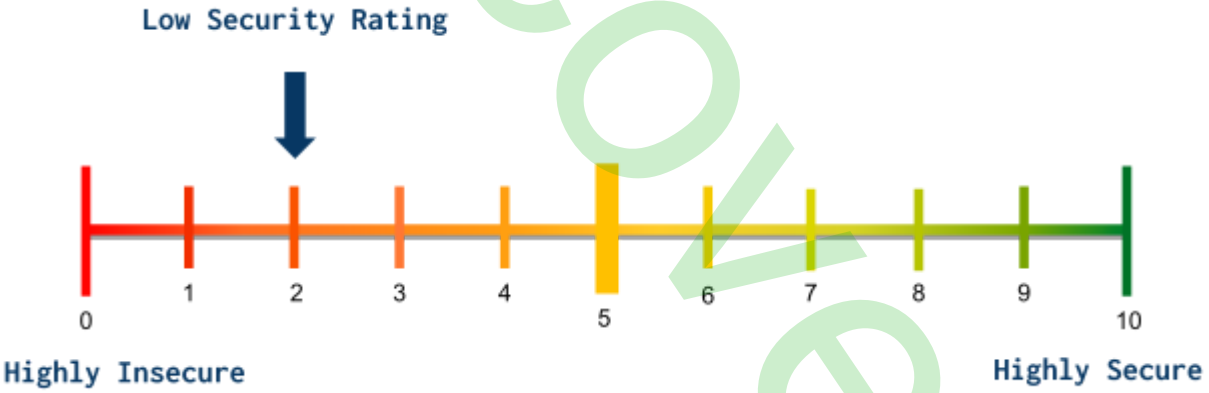
The application provides customers the ability to submit order requests, review design, leave feedback, etc.

The penetration test was conducted between **08.02.2021 - 26.02.2021**.

The penetration test has the following objectives:

- identify technical and functional vulnerabilities
- evaluate a severity level (ease of use, impact on information systems, etc.);
- make a prioritized list of recommendations to address identified weaknesses.

According to our research after performing the penetration testing, the security rating of the client's **android application** was identified as **Low**.



Scope

The following list of the information systems was the scope of the penetration testing.

#	Name	Description	Version
1.	__	Android	

Methodology

The testing methodology is based on generally accepted industry-wide approaches to perform penetration testing for mobile applications - Mobile Security Testing Guide (MSTG);

Application-level penetration tests include, at a minimum, checking for the following types of vulnerabilities:

- lack of binary protections;
- insecure data storage;
- unintended data leakage;
- client-side injection;
- weak encryption;
- implicit trust of all certificates;
- execution of activities using root;
- private key exposure;
- exposure of database parameters and SQL queries;
- insecure random number generator.

Severity Definition

The level of severity of each vulnerability is determined based on the potential impact of loss from successful exploitation as well as ease of exploitation, the existence of exploits in public access and other factors.

Severity	Description
High	High-level vulnerabilities are easy to exploit and may provide an attacker with complete control of the affected systems, leading to significant data loss or downtime. There are exploits or PoC available in public access.
Medium	Medium-level vulnerabilities are much harder to exploit and may not provide the same access to affected systems. Exploits or PoCs aren't available in public access. Exploitation provides only very limited access.
Low	Low-level vulnerabilities exploitation is extremely difficult, or impact is minimal.
Info	Information-level vulnerabilities provide an attacker with information that may assist them in conducting subsequent attacks against target information systems or against other information systems, which belong to an organization.

Summary of Findings

The table below shows the vulnerabilities and their severity. A total of **6 vulnerabilities** were found.

Title	Severity
Critical bug in money transfer	High
Personal data in logs	High
User enumeration	Medium
No certificate and public key pinning	Low
Http without headers	Info
Out of date library	Info

Based on our understanding of the application, as well as the nature of the vulnerabilities discovered, their exploitability, and the potential impact we have assessed the security rating of the client's **android application** as **Low**.

The client should pay special attention to the following vulnerabilities:

1. **Critical bug in money transfer.**

Key Findings

■■■■ Critical bug in money transfer

#1	Description
The application crashes, when money is transferred between two different accounts.	

Evidence

Steps to reproduce:

1. Log in application
2. Enter into money transfer
3. Get valet address with help QR-code
4. Input data
5. Press send button

Recommendations

- check transfer work on different versions of Android.

■■■■ Personal data in logs

#2 Description

Personal data can be stolen from application logs. Often Developers leave debugging information publicly. So any application with READ_LOGS permission can access those logs and can gain sensitive information through that.

Evidence

Steps to reproduce:

1. Perform pidcat

Request:

Recommendations

- turn off READ_LOGS permissions.

User enumeration

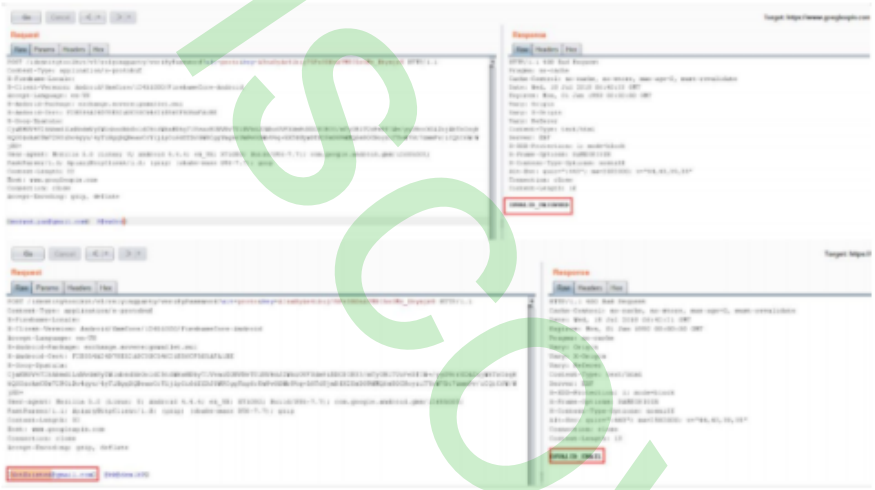
#3 Description

Authorization header contains both email address and authentication token. It was discovered that by sending existing and non-existing email addresses it is possible to enumerate valid users because of different responses. Before the token is checked, the application looks up if the email address belongs to a registered user. If the user is not registered, an error “User not found” occurs.

Evidence

Steps to reproduce:

1. Check response from existing and not existing user during authorization



Request:

Recommendations

- it is recommended to provide the same response irrespective of whether the password was incorrect or the username does not exist.

No certificate and public key pinning

#4	Description

There was no Certificate and Public Key Pinning found during the mobile application test. Absence of the mechanism makes it more convenient and faster to intercept and decrypt traffic between an application and a server. Pinning is the process of associating a host with their expected X509 certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or 'pinned' to the host.

Evidence

Steps to reproduce:

1. Configure the Burp Proxy listener
2. Configure your device to use the proxy
3. Test the configuration. If the traffic can be captured and decrypted the Pinning mechanism is not implemented

Recommendations

- implement Certificate and Public Key Pinning. For more details please visit https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

Http without headers

#5	Description
Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.(Cache-control: nostore, Pragma: no-cache)	

Recommendations
- Add the following headers: Cache-control: no-store Pragma: no-cache.

Out of date library

#6	Description

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information. As the java.util.Random class relies on a pseudorandom number generator, this class and relating java.lang.Math.random() method should not be used for security-critical applications or for protecting sensitive data java.util.Random. This package is flawed and produces predictable values for any given seed which are easily reproducible once the starting seed is identified.

java_source\o\uZ.jav[redacted]
java_source\okhttp3\[redacted]
java_source\[redacted]: 62
java_source\okhttp3\internal\ws\WebSocketWriter.java - Line: 19
java_source\o\[redacted] - Line: 13
java_source\com\google\[redacted]: 33
java_source\com\google\[redacted] Line: 34
java_source\o\[redacted]
[redacted]collect\[redacted]
java_source\[redacted]
java_source\[redacted] - Line: 17
java_source\o[redacted]- Line: 22
java_source\com\[redacted]

Recommendations

- use library java.security.SecureRandom, read more https://resources.infosecinstitute.com/randomnumber-generation-java

Appendix A. Automated Tools

Scope	Tools Used
Application Security	Drozer Xposed 3.1.5 MobSF Dex2Jar JD-GUI BurpSuite 1.7.30 Nmap Sqlmap VisualCodeGrepper SonarQube
Devices	Samsung Note 8 - Android 8.0 Lenovo A968 - Android 4.4.2 Motorola Z Force - Android 8.0 Motorola Droid Turbo 2 - Android 7.0 Motorola Droid Maxx - Android 4.4.4