DAMN VULNERABLE BANK(DVBA SETUP GUIDE)

Author: Rengoku

INTRODUCTION:

The setup will have 4 process

- 1. Building the app backend
- 2. Building the app frontend
- 3. Bypass Root + Ssl Pin + Frida
- 4. Decrypt Parameter Value

A.) BUILDING THE APP BACKEND

```
(rengoku@ kali)-[~/mobile/Damn-Vulnerable-Bank/BackendServer]
$ sudo docker-compose up ─-build -d
Creating network "backendserver_default" with the default driver
Building api
[+] Building 2.3s (10/10) FINISHED

$ [internal] load build definition from Dockerfile

$ † transferring dockerfile: 1768

$ [internal] load metadata for docker.io/library/node:10.19.0

$ [internal] load .dockerignore

$ † transferring context: 28

$ [1/5] FROM docker.io/library/node:10.19.0@sha256:816cfaee24dc2cea534e21d7f9c55f3b22c8bc6af61d8445f8d0178168ef3b28

$ [internal] load build context

$ † transferring context: 2.05kB

$ CACHED [2/5] WORKDIR /home/node/dvba

$ CACHED [3/5] COPY packages.json ./

$ ACAHED [4/5] RUN npm install

$ CACHED [5/5] COPY.

$ exporting to image

$ ⇒ exporting layers

$ ⇒ writing image sha256:04987d7b7cb332e456b76a7f2a24f47aa729dee74745aed13ad1901608314661

$ ⇒ naming to docker.io/library/backendserver_api
Creating backendserver_mysql_1 ... done
Creating backendserver_mysql_1 ... done

[* (rengoku@ kali)-[~/mobile/Damn-Vulnerable-Bank/BackendServer]

$ sudo docker-compose start
Starting mpsql ... done

Starting api ... done
```

Figure 1.1 building the app backend

- Download the DVBA repository and go to the backend directory which is BackendServer
- 2. Build the backend using docker with command "sudo docker-compose up -- build -d"
- 3. In the figure 1.1 above. Shown I have already build the backend server. So to start the server just use commands. "sudo docker-compose start"
- 4. COPY the backend IP which is on the eth0 network interface
- 5. DONE! you have successfully build the backend

B.) BUILDING THE APP FRONTEND

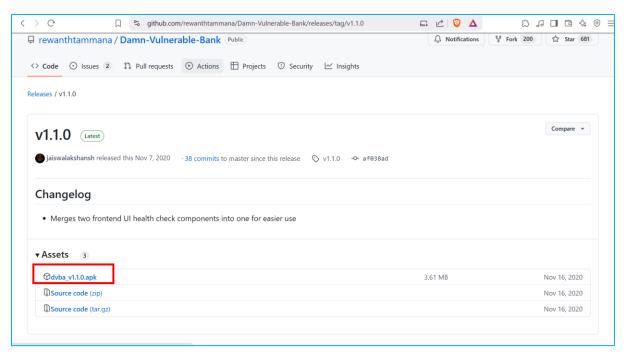


Figure 1.2 DVBA APK

- 1. Download the DVBA apk on your mobile device that want to be pentest.
- 2. Install The APP
- 3. To connect Front End with backend. At the server API section on the app insert below "http://BACKENDIP:3000". Check Health and go to sign up page.
- 4. If connection successful. You can create user account! if failed. It will say something went wrong.
- 5. DONE!

C.) BYPASS ROOT DETECTION + SSL PINNING + FRIDA DETECTION

```
PS C:\Users\rengo\OneDrive\Desktop\ASK-Pentest\projek MOBILE\frida scripts> adb shell ps | Select-String damnvulnerableb ank

u0_a178 6668 1343 5291304 132580 0 0 S com.app.damnvulnerablebank

PS C:\Users\rengo\OneDrive\Desktop\ASK-Pentest\projek MOBILE\frida scripts> |
```

Figure 1.3 Find DVBA Process ID

Figure 1.4 Inject Frida Scripts

- 1. Open Powershell
- 2. Run the commands "adb shell ps | Select-String damnvulnerablebank"
- 3. The process ID is 6668
- 4. Using The commands frida -U -p 6668 -l bypass_root_sslpin.js -l frida_detect.js You can bypass the app
- 5. '-p' 6668 is the process ID and '-l' is the script used
- 6. IF BYPASS FAILED. You won't be able to open the app at all

D.) DECRYPT PARAMETER VALUE

```
Pretty Raw Hex

Post /api/user/login HTTP/1.1
Content-Type: application/json; charset=utf-8
User-Agent: Dalvik/2.1.0 (Linux; U; Android 10; Nightmare Build/QQ3A.200805.001)
Host: 192.168.218.112:3000
Connection: keep-alive
Accept-Encoding: gzip, deflate, br
Content-Length: 73

"enc_data":"Gk8UCQwcCQAABFhTTBUEAwYVAhtFTU8RGxodEA4fBVhTTFFUXFJYFA==\n"
}
```

Figure 1.5 The App Encrypted Parameter Value

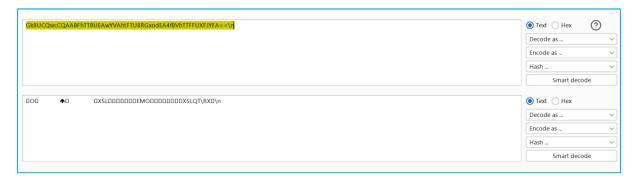


Figure 1.6 Decrypted using normal BASE64 Decoder

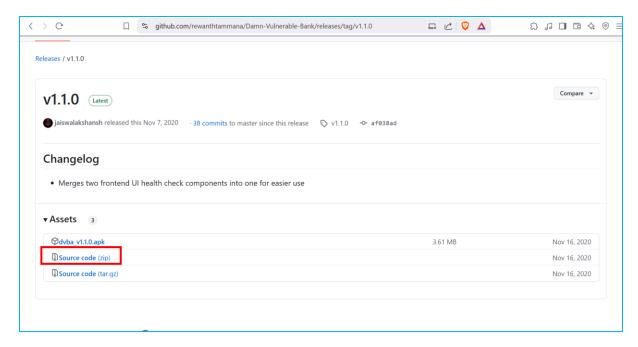


Figure 1.7 APP Source Code

```
PS C:\Users\rengo\OneDrive\Desktop\ASK-Pentest\projek MOBILE\frida scripts> python decryptEncrypt.py
Choose an option:
1. Encrypt text
2. Decrypt text
Enter choice (1/2): 2
Enter Base64-encoded text to decrypt: Gk8AGQoBEg8ZPhQcAwUEH0NAS19VUllDBw==\n

Decrypted text:
{"account_number":"1234"}
PS C:\Users\rengo\OneDrive\Desktop\ASK-Pentest\projek MOBILE\frida scripts>
```

Figure 1.8 Decrypted Parameter Value

- To Decrypt this first we must understand how it works. First read the backend Code of the app to understand it. The directory is in here "\Damn-Vulnerable-Bank-
 - 1.1.0\DamnVulnerableBank\app\src\main\java\com\app\damnvulnerablebank/EncryptDecrypt.java".
- 2. Everytime you want to alter the value of the parameter use the python script I have provided.

FINDINGS

```
2. Decrypt text
3. Exit
Enter choice (1/2/3): 2
Enter Base64-encoded text to decrypt: Gk8SDggaEhJPWwFLDQgFCENAXF5XTU8MHxodBgYIQ0BLJwkVCBMUCAJHMggTDAwcRyQFExUbTBpNTwUbHQ
PMxZDFAQDAKHXQykHMFXEAQgADCoGFQwDGxoLThMfDghLQkURDBMfBxpFWxZDGQYKAkHXQy0oPCk+K5AukDEzMzgvOSg6IiVPTVgMHBUPAkNAWFxRVEFDCR
gCNBUMFR9LVEVRXFFKWUxLQx4QFiQLFBIMBh9LVEULDBUbSRoVFAMCGxoLA0ELDghJDQgNGawUSUkFBAMEHAANDgAfGCUIDQQOGA8ONgASDA8ECESOBhVNEX
UeTLZDQUMJGAJFW808NDbrNTVNKDQ9JTUREDBWQUDAgOAgQACAALFAFNSRoAGGdNDQAZCgESDXk+FBBDBQQFAVVJDAIPCAcTCgcGEXQ+Gw0nNcBQQFSUHGw0DCB
MARQV4GERDFFR8LAWFEQSwo1jikPkFSLSshlDgtLkVRS158XINSTEDHQAICAMCFQgTCUUTDFZeUEJZXUtDXFBaBhxHUFBQWEUIBg9GeCUQkUOHWgAdAAGDU
DbAUSNCAUIQ0BLOSYZIZ4HXDomPjkzLyctJjUoJVhFTAITHw8VSIRWUltUVksdFg9+FRsdcOVbTTPLLWYSXQ0FDCRgCKqQeEhSOCOVbTyUbHQ9HFR8UFAoPEw
QJQRwGHEcCAg0PBABHRg8EFAwIDgIEAAgQMQYCDg4PBxv4DxgMGAwcQEEMFVobARBBXENWSx0WDD9bWCAgNCQ/NVogIDMuTQEYDAACBwQCEwgcDgQeAVPBD9
4FDU08ACA0EDhgPDjYAEgwPBAgJQgcDCA8FDweECAwTAzYPBAICFBQdMQkUAAMFGw5LAQwWChsBEQQJAVNJOCYtOCQpSUYjJCsgLyU6S15BXIZWRIxDQUMKCB
wGDAgYHxsdRVs2VklYYldSQQNULWE4IE010R1hMswcMDQkMMxpNTxILBUxdQyQvKSw8M0EkLy4mTgcDCA8FDwcECAwTEwwd80FFARNNOksBDAIZBhsJFTIPDw
QMAhMMTROLWkECwgZAASVGDIAGQoBEg8ZPhqcAwUEHwFWCQ8XER80DAwKB0hNNzsl0yIyTUk+LCgmNCEIVIZCWEISSEFLQkURDBMbBASTBB8SWFMIUFJcWU
paQkVQXEEVG05WXFxDVg8PCXIIPAcU

Decrypted text:
{"status":{"code":500,"message":"Internal Server Error"}, "data":{"name":"SequelizeDatabaseError", "parent":{"code":"WARN
DATA_TRUNCATED", "errno":1265, "sqlState":"01000", "sqlMessage":"Data truncated for column heneficiary_account_number', \account_number', \account_number
```

Figure 1.9 Error Based SQL Injection