COMP5618 - Applied Cybersecurity S2 2025 Mobile CTF

Individual Report

Matin Aliyev (541011493)

Challenge 1 - The serial Key is missing

- **Found flag - COMP5618{YOU_CRACKED_AN_APP}**

Description

- Decompiled the APK into challenge1 folder using "**apktool d -f challenge_1.apk -o challenge1**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.example.serial")
- Since launcher is in **com.example.serial.MainActivity**, I continued static analysis with investigating smali files related to MainActivity launcher: **MainActivity$1** and **MainActivity1**

```
.line 67
new-instance v7, Lcom/example/serial/MainActivity$1;

const-string v3, "AES/CBC/PKCS5Padding"

const-string v6, "3bcb972967a53b73a6f9700c82b07eaaaad7b2680a4b5150ffaece40915a9436"

move-object v0, v7

move-object v1, p0

invoke-direct/range {v0 .. v6}, Lcom/example/serial/MainActivity$1;-><init>(Lcom/example/serial/MainActivity;
Landroid/widget/TextView;Ljava/lang/String;Ljavax/crypto/SecretKey;Ljavax/crypto/spec/IvParameterSpec;Ljava/lang/String;)V

invoke-virtual {p1, v7}, Landroid/widget/Button;->setOnClickListener(Landroid/view/View$OnClickListener;)V
```

Figure: Hard-coded ciphertext in the smali code

The input is compared to the hardcoded ciphertext (encrypted using AES-CBC with key and IV).

```
.line 55
invoke-virtual {v0}, Lcom/example/serial/DBHelper;->getReadableDatabase()Landroid/database/sqlite/SQLiteDatabase;

move-result-object v0

const-string v1, "Select * from keys"

const/4 v3, 0x0
```

Figure: Retrieving keys from database in smali code

Code shows "keys" are selected from keys database, so I tried to view it to find a clue for ciphertext in **MainActivity.smali** file:
Command: sqlite3 database.db "SELECT * FROM keys;"

```
matin@RogMatin:/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge1/assets/databases$ sqlite3 database.db "SELECT
* FROM keys;"
50655368566D597133743677397A2443|2D4B6150645367566B59703373367639
33743677397A24432646294A404E6351|743677397A244326452948404D635166
33743677397A24432646294A404E6351|7638792F423F4528482B4B6250655368
38792F423F4428472B4B625065536856|7638792F423F4528482B4B6250655368
matin@RogMatin:/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge1/assets/databases$
```

Figure: Dumping exposed keys from database

- Then I tried decrypting the hardcoded ciphertext with the found keys and one of them gave the flag:
  Command: python script in the image

```
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge1> python
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from Crypto.Cipher import AES
>>> key = bytes.fromhex("33743677397A244326462946A404E6351")
>>> iv  = bytes.fromhex("7638792F423F4528482B4B6250655368")
>>> ct  = bytes.fromhex("3bcb972967a53b73a6f9700c82b07eaaaad7b2680a4b5150ffaece40915a9436")
>>> pt = AES.new(key, AES.MODE_CBC, iv).decrypt(ct)
>>> print(pt[:-pt[-1]].decode("utf-8"))
COMP5618{YOU_CRACKED_AN_APP}
>>>
```

Figure: Decrypting hard-coded ciphertext with found keys

Challenge 2 - Play with a little bit of help

- **Found flag - COMP5618{YOU_FOUND_BOTH_KNIGTS}**

Description

- I installed the challenge_2.apk on an emulator and opened the app for inspection. The **Help** screen showed two clues on top of the page:
  - "Look for the Knights"
  - "Assets will guide you"
- Then I decompiled the APK into challenge2 folder using "**apktool d -f challenge_2.apk -o challenge2**" command and started examining assets folder:
  - help_play-en.html contains what we saw above in the app.
  - help-enhtml contains "Look for the Black Knight".
- Since all clues pointed to assets folder and knights, I started inspecting knight images for any hidden files or information inside highres folder inside assets:
  Command:
  - **steghide extract -sf nw.jpg -p ""**
  - **sudo binwalk -e nb.png --run-as=root**

```
matin@RogMatin:/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge2/assets/highres$ steghide extract -sf nw.jpg -p ""
wrote extracted data to "wHITE_kNIGHT.txt".
matin@RogMatin:/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge2/assets/highres$ sudo binwalk -e  nb.png --run-as=root

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 128 x 128, 8-bit/color RGBA, non-interlaced
99            0x63            Zlib compressed data, best compression
2870          0xB36           Zip archive data, encrypted at least v2.0 to extract, compressed size: 57, uncompressed size: 34, name: hERE_Lies_tHE_Two_kNIGHTS.txt
3135          0xC3F           End of Zip archive, footer length: 22

matin@RogMatin:/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge2/assets/highres$
```

Figure: Content hidden in nb.png and nw.jpg files

wHITE_kNIGHT.txt contains a ciphertext:
f14a194adc3fbd3b23440f3731e9a15f065583e78aa3b16a7aea2396b5e8e28a
inside nb.png there is a zip folder that contains: hERE_Lies_tHE_Two_kNIGHTS.txt which is encrypted using AES.

- To crack the password, I tried using hashcat to compare the hash of the password with the rockyou.txt passwords' hash list:
  Commands:

- **./zip2hashcat "/mnt/c/Users/matin/Downloads/Assignment 3 Files/challenge2/assets/highres/B36.zip" > zip.hc**
- **hashcat -m 13600 -a 0 zip.hc "$ROCKYOU" --potfile-disable** (set $ROCKYOU to your path to run the command)

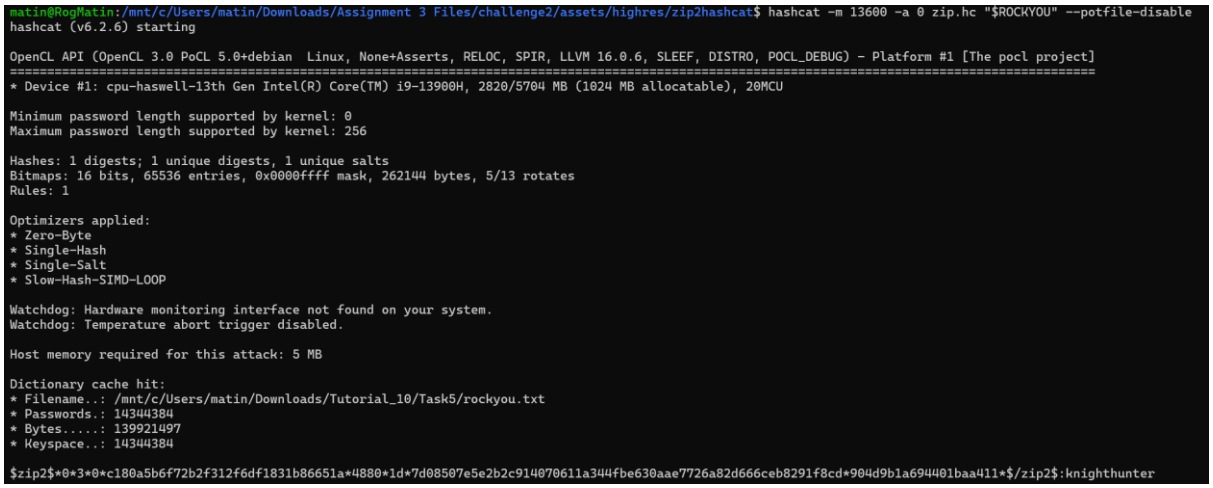


Figure: Breaking the encrypted file using Hashcat

- found password: **knighthunter**

content of hERE_Lies_tHE_Two_kNIGHTS.txt:

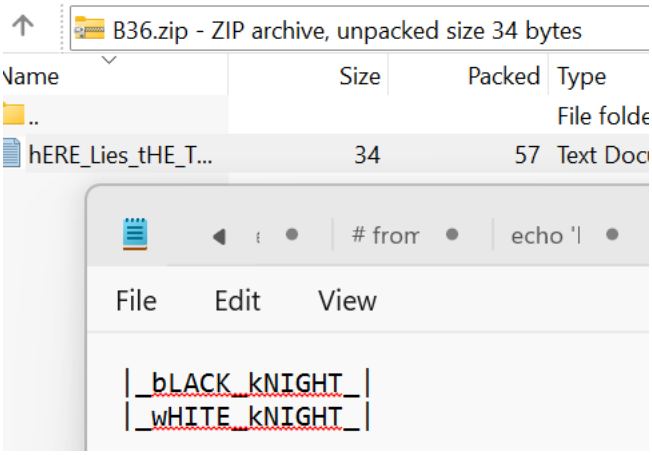


Figure: Content of decrypted file

- To decrypt the ciphertext we will need key and IV, so I tried converting weirdly written words (was one of the clue) to the ASCII bytes and tried using inputs as key and iv:
  Used Python Code:

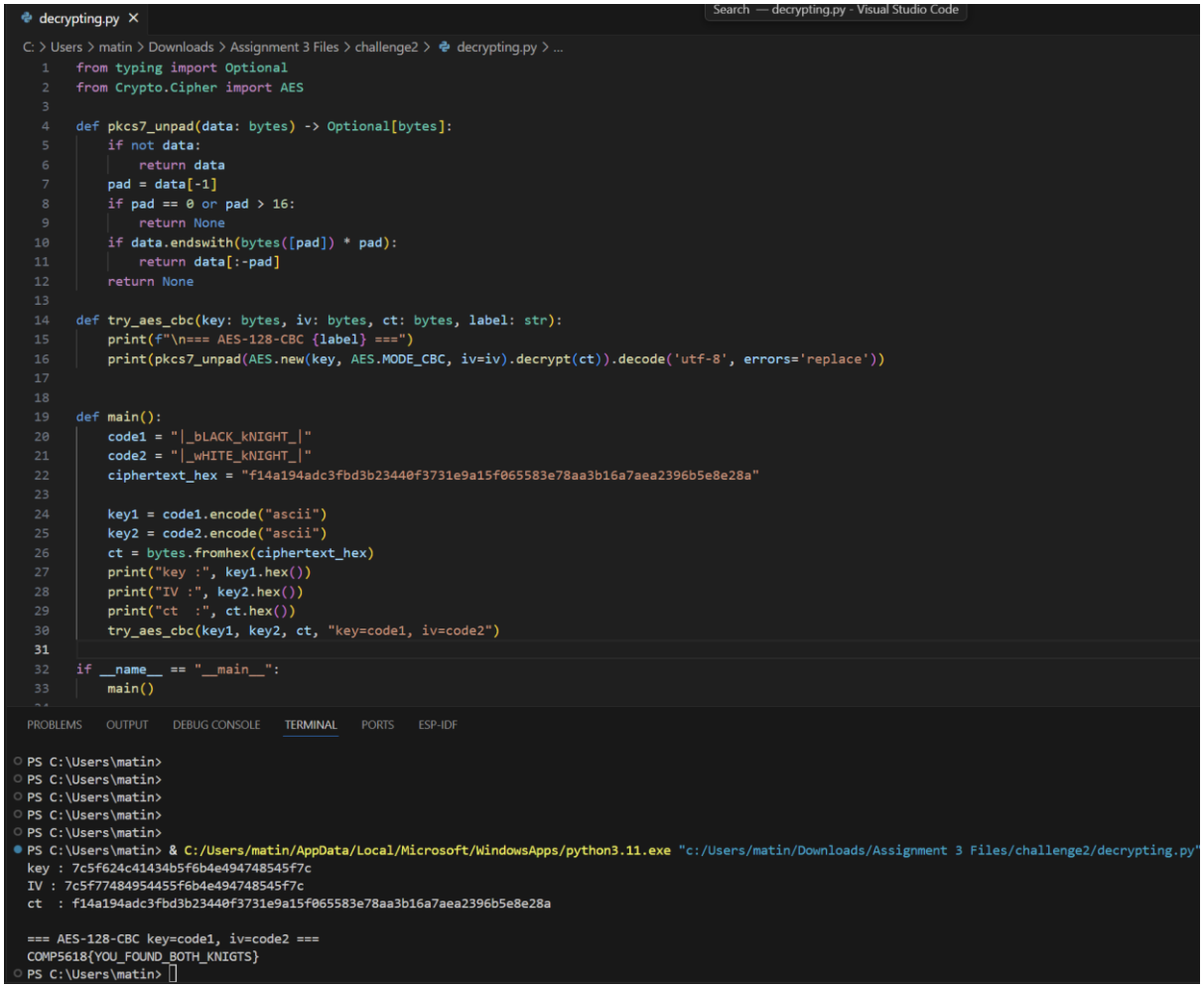


Figure: Python implementation to get the flag with known hints

Challenge 3 – Decompile it twice

- **Found flag - COMP5618{Jumbled_n_cOmpileD}**

Description

- Decompiled the APK into challenge3 folder using "**apktool d -f challenge_3.apk -o challenge3**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.dozingcatsoftware.bouncy")
- I started first investigating suspicious smali files linked to the main activity and API calls such as **com.dozingcatsoftware.ApiHelperJava com.dozingcatsoftware.bouncy.AboutActivity** and **com.dozingcatsoftware.bouncy.BouncyPreferences**:

```
invoke-direct {v0}, Lretrofit2/Retrofit$Builder;-><init>()V

invoke-static {}, Lcom/dozingcatsoftware/ApiHelperJava;->baseUrlFromJNI()Ljava/lang/String;

move-result-object v1

invoke-virtual {v0, v1}, Lretrofit2/Retrofit$Builder;->baseUrl(Ljava/lang/String;)Lretrofit2/Retrofit$Builder;

move-result-object v0
```

Figure: Usage of baseUrlFromJNI() method

```
# direct methods
.method static constructor <clinit>()V
    .locals 1

    const-string v0, "native-lib"

    .line 11
    invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V

    return-void
.end method
```

Figure: snippet showing loading of native-liv.so library during runtime which hints potentially hidden flag in that library

- I started checking what JNI returns during runtime using Frida:
  Commands:
  (install Frida and corresponding server image to run it)
  - **adb root**
  - **adb push frida-server-17.4.0-android-x86_64 /data/local/tmp/frida-server**
  - **adb shell "chmod 755 /data/local/tmp/frida-server"**
  - **adb shell "sh -c '/data/local/tmp/frida-server >/data/local/tmp/frida.log 2>&1 < /dev/null &'"**
  - **adb shell "pidof frida-server"**
  - **adb forward tcp:27042 tcp:27042**

Figure: Implemented JavaScript code to see what function returns during runtime

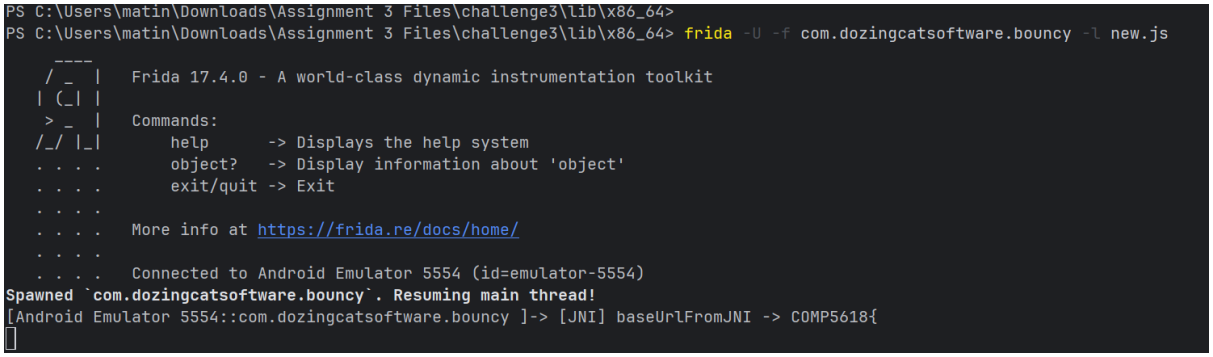


Figure: Frida shows output of function during runtime

- The beginning of the flag was returned, however the rest of it was still hidden. Since the challenge name says, “decompile it twice”, this time I tried decompiling the **libnative-lib.so** to check for further hints. I used Ghidra to do:

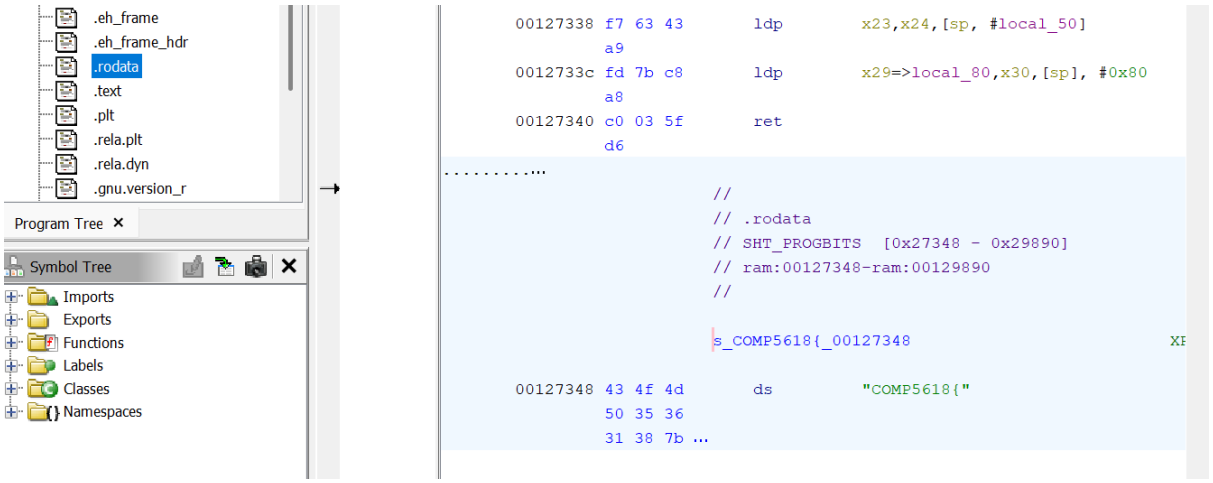


Figure: rodata file contains the beginning of the flag COMP5618{ as in the snippet

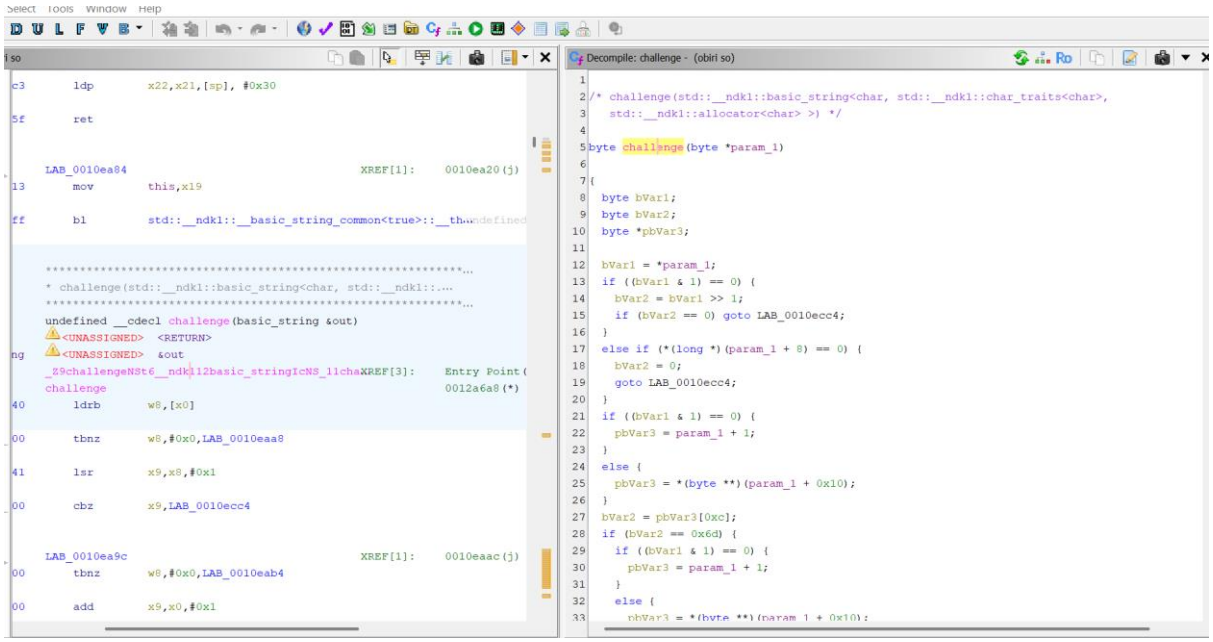


Figure: the challenge function and second decompile

- Under Functions/challenge, the rest of the flag is given as bytes that could be found after decompiling as it was hinted in the challenge name:

0x4a	J	0x65	e	0x63	c	0x6c	l
0x75	u	0x64	d	0x4f	O	0x65	e
0x6d	m	0x5f	_	0x6d	m	0x44	D
0x42	B	0x6e	n	0x70	p	0x7d	}
0x6c	l	0x5f	_	0x69	i		

Figure: ASCII codes and human-readable versions

- So, the final completed flag becomes **COMP5618{Jumbled_n_cOmpileD}**

Challenge 4 – What is the combination

- **Found flag - COMP5618{gEt_oVeR_hErE}**

Description

- Decompiled the APK into challenge_4 folder using "**apktool d -f challenge_4.apk -o challenge_4**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.example.combination")
- I started investigating ApiHelper1.smali, ApiHelper2.smali and MainActivity.smali file for any hints:

```
new-instance v0, Ljava/net/URL;

invoke-direct {v0, p1}, Ljava/net/URL;-><init>(Ljava/lang/String;)V

.line 119
new-instance p1, Ljava/util/Scanner;

invoke-virtual {v0}, Ljava/net/URL;->openStream()Ljava/io/InputStream;

move-result-object v0
```

Figure: URL building part in MainActivity.smali file

```
:array_0
.array-data 1
    0x4bt
    0x61t
    0x50t
    0x64t
    0x52t
    0x67t
    0x55t
    0x6bt
    0x58t
    0x70t
    0x32t
    0x73t
    0x35t
    0x76t
    0x38t
    0x79t
.end array-data
```

```
:array_1
.array-data 1
    0x51t
    0x66t
    0x54t
    0x68t
    0x57t
    0x6dt
    0x5at
    0x71t
    0x34t
    0x74t
    0x37t
    0x77t
    0x21t
    0x7at
    0x25t
    0x43t
.end array-data
```

Figure: ASCII array bytes that decodes to key and iv

```
invoke-static {}, Lcom/example/combination/ApiHelper1;->getthis1()Ljava/lang/String;
move-result-object v1
invoke-virtual {v0, v1}, Lretrofit2/Retrofit$Builder;->baseUrl(Ljava/lang/String;)Lretrofit2/Retrofit$Builder;
move-result-object v0
invoke-direct {v0}, Lretrofit2/Retrofit$Builder;-><init>()V
invoke-static {}, Lcom/example/combination/ApiHelper2;->getthis2()Ljava/lang/String;
move-result-object v1
```

Figure: getthis methods in ApiHelper classes to get base64 strings

- After converting ASCII bytes into readable form, they became:
    - "KaPdRgUkXp2s5v8y"
    - "QfThWmZq4t7w!z%C"

  Which corresponds to key and iv we can use.
- To see what base64 strings ApiHelper methods actually are, I used Frida and hooked a JavaScript code to see the runtime fetch:
  Commands:
  (install Frida and corresponding server image to run it)
    - **adb root**
    - **adb push frida-server-17.4.0-android-x86_64 /data/local/tmp/frida-server**
    - **adb shell "chmod 755 /data/local/tmp/frida-server"**
    - **adb shell "sh -c '/data/local/tmp/frida-server >/data/local/tmp/frida.log 2>&1 < /dev/null &'"**
    - **adb shell "pidof frida-server"**
    - **adb forward tcp:27042 tcp:27042**

```
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge_4> frida -U -f com.example.combination -l hook_inject.js
     ____
    . . . .       exit/quit -> Exit
    . . . .
    . . . .   More info at https://frida.re/docs/home/
    . . . .
    . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `com.example.combination`. Resuming main thread!
[Android Emulator 5554::com.example.combination ]-> [*] OkHttp interceptor installed (builder + ctor).
[*] pass+net hooks armed — any combo should be accepted, and decrypted text will be printed.
[+] getthis1 -> xWd/yRAaEb6ufEnyCvvZ94JEvDaf5gxbFTtjxnUnKT9xr+RpisuB/TKzHXVoQIWQ
[+] getthis2 -> xWd/yRAaEb6ufEnyCvvZ93+C1GNEEG/CSXFcOYJeZ+zr4yrWuF4q2dQSsWhhtFQZ
[+] getthis1 -> xWd/yRAaEb6ufEnyCvvZ94JEvDaf5gxbFTtjxnUnKT9xr+RpisuB/TKzHXVoQIWQ
[+] getthis2 -> xWd/yRAaEb6ufEnyCvvZ93+C1GNEEG/CSXFcOYJeZ+zr4yrWuF4q2dQSsWhhtFQZ
[Android Emulator 5554::com.example.combination ]->
[Android Emulator 5554::com.example.combination ]-> exit

Thank you for using Frida!
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge_4>
```

Figure: Obtaining runtime base64 strings using JavaScript code to see what functions return

hook_inject.js

C: > Users > matin > Downloads > Assignment 3 Files > challenge_4 > hook_inject.js > ...

```
1   Java.perform(function () {
2     try {
3       const Api1 = Java.use('com.example.combination.ApiHelper1');
4       const Api2 = Java.use('com.example.combination.ApiHelper2');
5       console.log('[+] getthis1 -> ' + Api1.getthis1());
6       console.log('[+] getthis2 -> ' + Api2.getthis2());
7     } catch (e) {
8       console.log('[-] Couldn't call helpers: ' + e);
9     }
10  });
11
```

Figure: Used JavaScript code

- After obtaining ciphertexts I decrypted them in Powershell and obtained following urls:
  - https://www.pastebin.com/ZGtmighT
  - https://www.pastebin.com/GeRXQmp2

```
PS C:\Users\matin> function Decrypt-B64([string]$b64) {
>>   $key = "KaPdRgUkXp2s5v8y"
>>   $iv  = "QfThWmZq4t7w!z%C"
>>   $aes = [System.Security.Cryptography.Aes]::Create()
>>   $aes.Mode = 'CBC'; $aes.Padding = 'PKCS7'
>>   $aes.Key = [Text.Encoding]::UTF8.GetBytes($key)
>>   $aes.IV  = [Text.Encoding]::UTF8.GetBytes($iv)
>>   $ct = [Convert]::FromBase64String($b64)
>>   $pt = $aes.CreateDecryptor().TransformFinalBlock($ct,0,$ct.Length)
>>   [Text.Encoding]::UTF8.GetString($pt)
>> }
PS C:\Users\matin>
PS C:\Users\matin> $enc1 = 'xWd/yRAaEb6ufEnyCvvZ94JEvDaf5gxbFTtjxnUnKT9xr+RpisuB/TKzHXVoQIWQ'
PS C:\Users\matin> $enc2 = 'xWd/yRAaEb6ufEnyCvvZ93+C1GNEEG/CSXFcOYJeZ+zr4yrWuF4q2dQSsWhhtFQZ'
PS C:\Users\matin>
PS C:\Users\matin> $u1 = Decrypt-B64 $enc1
PS C:\Users\matin> $u2 = Decrypt-B64 $enc2
PS C:\Users\matin> "$u1"
https://www.pastebin.com/ZGtmighT
PS C:\Users\matin> "$u2"
https://www.pastebin.com/GeRXQmp2
PS C:\Users\matin>
```

Figure: Decrypted URLs

- First URL gave sequences to get flag in app: sstctctsssstcstcsscstttssstc
- Second URL gave the actual flag: COMP5618{get_ovEr_hErE}

Challenge 5 – Salted and hashed

- **COMP5618{ALWAYS_ADD_SALT_BEFORE_HASHING}**

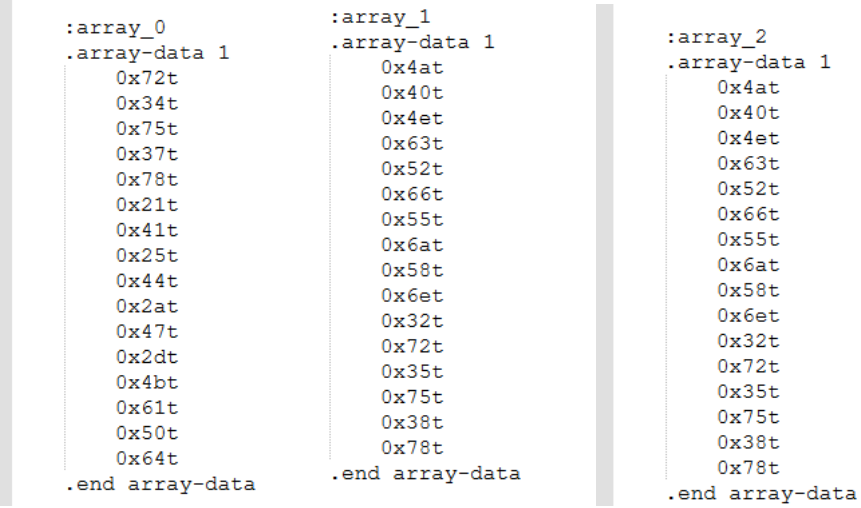
Description

- Decompiled the APK into challenge5 folder using "**apktool d -f challenge_5.apk -o challenge5**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.example.logintosee")
- Since launcher is in **com.example.logintosee.MainActivity**, I continued static analysis with investigating smali files related to MainActivity launcher: **MainActivity$1** and **MainActivity1**

```
const/16 v0, 0x10

new-array v3, v0, [B

.line 48
fill-array-data v3, :array_0

const-string v1, "AES/CBC/PKCS5Padding"

new-array v5, v0, [B

.line 53
fill-array-data v5, :array_1

.line 54
new-instance v6, Ljavax/crypto/spec/SecretKeySpec;

const/4 v7, 0x0

const-string v8, "AES"

invoke-direct {v6, v5, v7, v0, v8}, Ljavax/crypto/spec/SecretKeySpec;-><init>([BIILjava/lang/String;)V

new-array v0, v0, [B

.line 56
fill-array-data v0, :array_2

.line 57
new-instance v5, Ljavax/crypto/spec/IvParameterSpec;

invoke-direct {v5, v0}, Ljavax/crypto/spec/IvParameterSpec;-><init>([B)V

:try_start_0
const-string v0, "CMEVTF/B+oFCNoiZ8mgrL1V951OD43LK0FN9MHnfiUk61bhpZ3Di8XtmP6Ghfv3v"

.line 61
invoke-static {v1, v0, v6, v5}, Lcom/example/logintosee/MainActivity;->decrypt(Ljava/lang/String;Ljava/lan
```

Figure: Decryption logic and the hard coded base64 string in smali code

Full decryption details are given, it uses AES-CBC scheme for encryption with accepting array_0, array_1 and array_2 for salt, key and iv respectively.



```
:array_0
.array-data 1
    0x72t
    0x34t
    0x75t
    0x37t
    0x78t
    0x21t
    0x41t
    0x25t
    0x44t
    0x2at
    0x47t
    0x2dt
    0x4bt
    0x61t
    0x50t
    0x64t
.end array-data
```

```
:array_1
.array-data 1
    0x4at
    0x40t
    0x4et
    0x63t
    0x52t
    0x66t
    0x55t
    0x6at
    0x58t
    0x6et
    0x32t
    0x72t
    0x35t
    0x75t
    0x38t
    0x78t
.end array-data
```

```
:array_2
.array-data 1
    0x4at
    0x40t
    0x4et
    0x63t
    0x52t
    0x66t
    0x55t
    0x6at
    0x58t
    0x6et
    0x32t
    0x72t
    0x35t
    0x75t
    0x38t
    0x78t
.end array-data
```

Figure: Hard-coded array bytes that are used for decryption

The hardcoded bytes of arrays are given in the end of the file code which I used to I decoded to the ASCII format to get the actual salt, key and IV.
There is also a constant base64 string in the first picture:
CMEVTF/B+oFCNoiZ8mgrL1V951OD43LK0FN9MHnfiUk61bhpZ3Di8XtmP6Ghfv3v
So, when input is received, it is always equal to “admin” and password is checked with hardcoded hex value
I decoded this base64 string to get the actual ciphertext and tied everything together to get the flag:
Used Python Code:

```python
B64 = "CMEVTF/B+oFCNoiZ8mgrL1V951OD43LK0FN9MHnfiUk61bhpZ3Di8XtmP6Ghfv3v"

def parse_smali_array(block: str) -> bytes:
    # grab all 0x.. (optionally ending with 't'), turn into integers
    hexes = re.findall(r'0x([0-9a-fA-F]+)t?', block)
    return bytes(int(h, 16) for h in hexes)

salt = parse_smali_array(ARRAY_0_SMALI)
key  = parse_smali_array(ARRAY_1_SMALI)
iv   = parse_smali_array(ARRAY_2_SMALI)

print("salt:", salt)
print("key :", key)
print("iv :", iv)


ct = base64.b64decode(B64)
print("ct :",ct)
pt = unpad(AES.new(key, AES.MODE_CBC, iv=iv).decrypt(ct), 16)
print("FLAG:", pt.decode())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ESP-IDF

PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5> & C:/Users/matin/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c
 3 Files/challenge5/challenge5code.py"
salt: b'r4u7x!A%D*G-KaPd'
key : b'J@NcRfUjXn2r5u8x'
iv : b'J@NcRfUjXn2r5u8x'
ct : b'\x08\xc1\x15L_\xc1\xfa\x81B6\x88\x99\xf2h+/U}\xe7S\x83\xe3r\xca\xd0S}0y\xdf\x89I:\xd5\xb8igp\xe2\xf1{f?\xa1\xa1~\xfd\xef'
FLAG: COMP5618{ALWAYS_ADD_SALT_BEFORE_HASHING}
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5>
```

Figure: Python implementation of decrypting the flag with found hints

Challenge 6 – Can you open it?

- **COMP5618{El_Rata_Alada}**

Description

- I installed x86 image to be compatible with the given **Pixel_5_API_28** (Google APIs x86) device. After installing the Image and starting the device, the PIN screen popped up that required the PIN to access. First, I thought flag will be given once I unlock the lock screen:
  Commands:
  - **adb root**
  - **adb wait-for-device**
  - **adb shell id**
  - **adb shell "rm -f /data/system/locksettings.db* /data/system/locksettings_sp.blob /data/system/gatekeeper.*"**
  - **adb reboot**

```
PS C:\Users\matin\AndroidStudioProjects\whatever> & $adb shell id
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(adb),
:su:s0
PS C:\Users\matin\AndroidStudioProjects\whatever>
```

Figure: Rooting the emulator image

I tried rooting the phone to get the root privileges so I can remove the lock screen to see what is inside.

Once inside, I started inspecting for the any hint that is hidden and found some suspicious data in Messages section:

- 73357638792F423F4528482B4B625065
- 645367556B58703273357638792F423F
- 55697538a3168769b520b63bdc07f71351894f4b9b11bfc251ad3131e761edcf

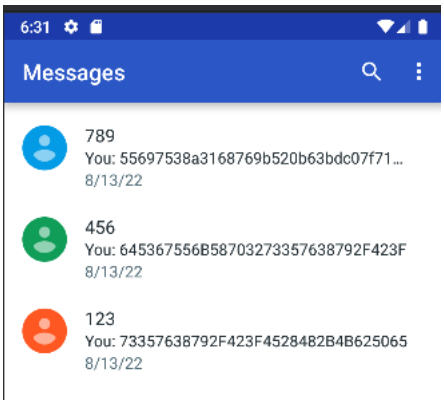


Figure: Findings in the Message section

- The found data is 2 16-byte long strings which hints for the key or IV or AES and a hex value which is most probably a ciphertext.
- I put every finding together and tried to decrypt the ciphertext to get the flag:

```
challenge6_code.py > ...
1   from Crypto.Cipher import AES
2   from Crypto.Util.Padding import unpad
3
4   key = bytes.fromhex("73357638792F423F4528482B4B625065")
5   iv  = bytes.fromhex("645367556B58703273357638792F423F")
6   ct  = bytes.fromhex("55697538a3168769b520b63bdc07f71351894f4b9b11bfc251ad3131e761edcf")
7
8   pt = unpad(AES.new(key, AES.MODE_CBC, iv=iv).decrypt(ct), 16)
9   print(pt.decode())
10

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ESP-IDF

PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5> & C:/Users/matin/AppData/Local/Microsoft/Windo
wnloads/Assignment 3 Files/challenge5/challenge6_code.py"
COMP5618{El_Rata_Alada}
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge5>
```

Figure: Python implementation of solving the flag

Challenge 7 – Where is the key?

- **Found flag - COMP5618{Brooks_was_here}**

Description

- I installed the challenge_2.apk on an emulator and opened the app for inspection. There was a likely base64 string and a null button. The button was doing nothing and when I decoded base64 string it looked like an RSA key since it was 256 bytes. It felt like I needed to inspect decompiled files for further clues:

Figure: UI view

- Decompiled the APK into challenge7 folder using "**apktool d -f challenge_7.apk -o challenge7**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.example.opentosee" and strings.xml for the given text:

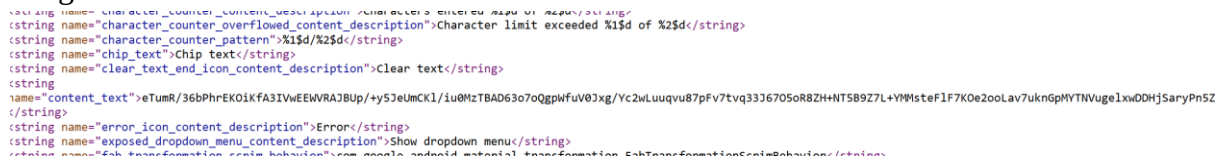


Figure: Strings.xml view

- Since launcher is in **com.example.opentosee.MainActivity**, I continued static analysis with investigating smali files related to MainActivity launcher: **MainActivity$1** and **MainActivity1**
- The MainActivity code does setContentView() but there is no click button handler implemented, so it basically does nothing. Since the application logic gives nothing and there was nothing to be found on UI format, I started inspecting alternative folders that we derived during decompile. Also, since the given key length is 256 bytes when decoded and is not human-readable, it clued that the given key is RSA-

2048 key. Since the other half of the key (public key) is needed and it was nowhere to be found in assets, res or smali files, I checked the certificate folder since each app supplies a signed certificate with the public RSA key and tried to use it to combine with the given key and see if it decrypts anything meaningful:

Commands:

```
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF> @'
>> eTumR/36bPhrEKOiKfA3IVwEEWVRAJBUp/+y5JeUmCKl/iu0MzTBAD63o7oQgpWfuV0Jxg/Yc2wLuuqvu87pFv7tvq33J67O5oR8ZH+NT5B9Z7L+YMMsteFlF7KOe2ooLav7uknGpMYTNVugelxwDDHjSaryPn5ZrVnyWb4Ddalnr3wP
dKYd77M714pgBzZiqhh4uEzzWIuufb3csFAADI8UE6H+7WDPmN3mYEUKl7oNDSQX/JHNxg/2VURy5NYjnRk1dHEZ5F0oS6VsFBI1uGHTJ0YTn5bypP5oXpq8U+Gp/2hVmF09c9j01Rvl6mFf616U8gaGrz7fdMjtmgsPA==
>> '@ | Set-Content content.b64
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF> certutil -decode content.b64 blob.bin > $null
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF> openssl pkcs7 -inform DER -in CERT.RSA -print_certs -out certs.pem
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF> openssl pkeyutl -verifyrecover -certin -inkey certs.pem -pkeyopt rsa_padding_mode:pkcs1 -in blob.bin
COMP5618{Brooks_was_here}
```

Figure: OpenSSL Powershell implementation

```python
import base64, subprocess
from pathlib import Path

B64 = """eTumR/36bPhrEKOiKfA3IVwEEWVRAJBUp/+y5JeUmCKl/iu0MzTBAD63o7oQgpWfuV0Jxg/Yc2wLuuqvu87pFv7tvq33J67O5oR8ZH+NT5B9Z7L+YMMsteFlF7KOe2ooL
HERE = Path(__file__).resolve().parent
cert_rsa = HERE / "CERT.RSA"
work = HERE / "work"
work.mkdir(exist_ok=True)

(work/"blob.bin").write_bytes(base64.b64decode(B64))

subprocess.check_call(["openssl","pkcs7","-inform","DER","-in",str(cert_rsa),"-print_certs","-out",str(work/"cert.pem")])
subprocess.check_call(["openssl","x509","-in",str(work/"cert.pem"),"-pubkey","-noout","-out",str(work/"pub.pem")])

out = subprocess.check_output([
    "openssl","pkeyutl","-verifyrecover","-pubin",
    "-inkey", str(work/"pub.pem"),
    "-pkeyopt","rsa_padding_mode:pkcs1",
    "-in", str(work/"blob.bin")
])

print(out.decode().strip())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ESP-IDF                                                    Python

PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF> & C:/Users/matin/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Use
wnloads/Assignment 3 Files/challenge7/original/META-INF/challenge_7.py"
COMP5618{Brooks_was_here}
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge7\original\META-INF>
```

Figure: Python implementation

Challenge 8 – Catch them all

- **COMP5618{YOU_FOUND_THE_BOX}**

Description

- I installed the challenge_8.apk on an emulator and opened the app for inspection. There was a “CLICK HERE” button it seems to perform no event on UI side.

Figure: UI view of Seven(application)

- Decompiled the APK into challenge8 folder using "**apktool d -f challenge_8.apk -o challenge8**" command and started examining AndroidManifest.xml to find out package and the launcher. (package="com.example.seven")
- Since launcher is in **com.example.seven.MainActivity**, I continued static analysis with investigating smali files related to MainActivity launcher: **MainActivity$1** and **MainActivity1:**
  There are two arrays given in raw bytes in the end of **MainActivity1** such that first array becomes AES key and the second array becomes IV during runtime decryption:

```
:array_0
.array-data 1
    0x70t
    0x32t
    0x73t
    0x35t
    0x76t
    0x38t
    0x79t
    0x2ft
    0x42t
    0x3ft
    0x45t
    0x28t
    0x48t
    0x2bt
    0x4dt
    0x62t
.end array-data
```

```
:array_1
.array-data 1
    0x53t
    0x68t
    0x56t
    0x6dt
    0x59t
    0x71t
    0x33t
    0x74t
    0x36t
    0x77t
    0x39t
    0x7at
    0x24t
    0x43t
    0x26t
    0x46t
.end array-data
```

```
invoke-static {}, Lcom/example/seven/ApiHelper;->baseUrlFromJNI()Ljava/lang/String;

move-result-object v1

iget-object v2, p0, Lcom/example/seven/MainActivity$1;->val$k:Ljavax/crypto/SecretKey;

iget-object v3, p0, Lcom/example/seven/MainActivity$1;->val$vi_para:Ljavax/crypto/spec/IvParameterSpec;

invoke-static {v0, v1, v2, v3}, Lcom/example/seven/MainActivity;->decrypt(Ljava/lang/String;Ljava/lang/Strir
```

Figure: Decryption of first URL from JNI

```
invoke-static {}, Lcom/example/seven/ApiHelper1;->baseUrlFromJNI1()Ljava/lang/String;
move-result-object v1
iget-object v2, p0, Lcom/example/seven/MainActivity$1;->val$k:Ljavax/crypto/SecretKey;
iget-object v3, p0, Lcom/example/seven/MainActivity$1;->val$vi_para:Ljavax/crypto/spec/IvParameterSpec;
invoke-static {v0, v1, v2, v3}, Lcom/example/seven/MainActivity;->decrypt(Ljava/lang/String;Ljava/lang/S
```

Figure: Decryption of second URL from JNI

```
invoke-static {}, Lcom/example/seven/ApiHelper2;->baseUrlFromJNI2()Ljava/lang/String;
move-result-object v1
iget-object v2, p0, Lcom/example/seven/MainActivity$1;->val$k:Ljavax/crypto/SecretKey;
iget-object v3, p0, Lcom/example/seven/MainActivity$1;->val$vi_para:Ljavax/crypto/spec/IvParameterSpec;
invoke-static {v0, v1, v2, v3}, Lcom/example/seven/MainActivity;->decrypt(Ljava/lang/String;Ljava/lang/String;Lj
```

Figure: Decryption of third URL from JNI

```
invoke-static {}, Lcom/example/seven/ApiHelper3;->baseUrlFromJNI3()Ljava/lang/String;
move-result-object v1
iget-object v2, p0, Lcom/example/seven/MainActivity$1;->val$k:Ljavax/crypto/SecretKey;
iget-object v3, p0, Lcom/example/seven/MainActivity$1;->val$vi_para:Ljavax/crypto/spec/IvParameterSpec;
invoke-static {v0, v1, v2, v3}, Lcom/example/seven/MainActivity;->decrypt(Ljava/lang/String;Ljava/lang/Str
```

Figure: Decryption of fourth URL from JNI

```
invoke-static {}, Lcom/example/seven/ApiHelper4;->baseUrlFromJNI4()Ljava/lang/String;
move-result-object v1
iget-object v2, p0, Lcom/example/seven/MainActivity$1;->val$k:Ljavax/crypto/SecretKey;
iget-object p0, p0, Lcom/example/seven/MainActivity$1;->val$vi_para:Ljavax/crypto/spec/IvParameterSpec;
invoke-static {v0, v1, v2, p0}, Lcom/example/seven/MainActivity;->decrypt(Ljava/lang/String;Ljava/lang/Strin
```

Figure: Decryption of fifth URL from JNI

- Since The URLs are decrypted during runtime, I used Frida to hook a JavaScript code to the running application to retrieve the decrypted URLs:
  Commands:
  (install Frida and corresponding server image to run it)
  - **adb root**
  - **adb push frida-server-17.4.0-android-x86_64 /data/local/tmp/frida-server**
  - **adb shell "chmod 755 /data/local/tmp/frida-server"**
  - **adb shell "sh -c '/data/local/tmp/frida-server >/data/local/tmp/frida.log 2>&1 < /dev/null &'"**
  - **adb shell "pidof frida-server"**

- **adb forward tcp:27042 tcp:27042**

```
Thank you for using Frida!
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge8> frida -U -f com.example.seven -l .\hook_decrypt.js
     ____
    / _  |   Frida 17.4.0 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `com.example.seven`. Resuming main thread!
[Android Emulator 5554::com.example.seven ]-> [ok] hooked MainActivity.decrypt(...)
[ok] hooked MainActivity.getdata(String)
[decrypt] alg=AES/CBC/PKCS5Padding enc=APE31ZTb0JPhUVUQxYIOxjMezyQ3nyxUuWdo4sPfxNUBvEjZtbIEC7b0C8LYwiZt -> https://pastebin.com/raw/jiXZKq3b
[getdata] https://pastebin.com/raw/jiXZKq3b
[decrypt] alg=AES/CBC/PKCS5Padding enc=APE31ZTb0JPhUVUQxYIOxqUnmhmsZxAYmGGrSo7A1C3E/Nn04vMsgB7hfmwG3RGD -> https://pastebin.com/raw/JFR8pKQr
[getdata] https://pastebin.com/raw/JFR8pKQr
[decrypt] alg=AES/CBC/PKCS5Padding enc=APE31ZTb0JPhUVUQxYIOxspnWdh1A84iCcyAaMxbTkWic/15AIha02B8yyFGQsRW -> https://pastebin.com/raw/ZkJcM4v5
[getdata] https://pastebin.com/raw/ZkJcM4v5
[decrypt] alg=AES/CBC/PKCS5Padding enc=APE31ZTb0JPhUVUQxYIOxtbnaGS2xA53vOf3A0wWZiQnUhOhBeaaad8Ch7q9wz78 -> https://pastebin.com/raw/CxqJ49GT
[getdata] https://pastebin.com/raw/CxqJ49GT
[decrypt] alg=AES/CBC/PKCS5Padding enc=APE31ZTb0JPhUVUQxYIOxkMvsoKfUW+KT8vNpMt04E85Y2yS8+ub3SRHpZ4hO1c1 -> https://pastebin.com/raw/Xr0uxDLJ
[getdata] https://pastebin.com/raw/Xr0uxDLJ
[Android Emulator 5554::com.example.seven ]->
```

Figure: Frida JavaScript code hook to catch the decrypted URLs during runtime

hook_decrypt.js

C: > Users > matin > Downloads > Assignment 3 Files > challenge8 > hook_decrypt.js > Java.perform() callback > implementation

```
Java.perform(function () {
  function log(tag, msg){ console.log("[" + tag + "] " + msg); }

  try {
    const Main = Java.use('com.example.seven.MainActivity');
    Main.decrypt.overload('java.lang.String','java.lang.String','javax.crypto.SecretKey','javax.crypto.spec.IvParameterSpec')
      .implementation = function (alg, enc, key, iv) {
        const out = this.decrypt(alg, enc, key, iv);
        log('decrypt', 'alg=' + alg + ' enc=' + enc + ' -> ' + out);
        return out;
      };
    log('ok', 'hooked MainActivity.decrypt(...)');
  } catch (e) { log('warn', 'decrypt hook failed: ' + e); }

  try {
    const Main = Java.use('com.example.seven.MainActivity');
    Main.getdata.overload('java.lang.String').implementation = function (url) {
      log('getdata', url);
      return this.getdata(url);
    };
    log('ok', 'hooked MainActivity.getdata(String)');
  } catch (e) { log('warn', 'getdata hook failed: ' + e); }

});

```

Figure: JavaScript code that is used to hook the running application for decrypting URLs

URLs:

- https://pastebin.com/raw/jiXZKq3b
- https://pastebin.com/raw/JFR8pKQr
- https://pastebin.com/raw/ZkJcM4v5
- https://pastebin.com/raw/CxqJ49GT
- https://pastebin.com/raw/Xr0uxDLJ

- Combining partial ciphertexts we obtained from each URL logically: “_SXIUQ_ “, “5618”, “CVL_OXL}”, “LCTC”, “{HCB” becomes LCTC5618{HCB_SXIUQ_CVL_OXL}
- Since flag starts with COMP5618{…}, it suggests that the COMP became LCTC. The only logical encryption methods to be considered are letter shifting schemes since input and output size is same and no key is provided.

The shifts suggest Vigenère rather than Caesar since O and P ended up with the same letter C:

- C (3) – L (12) Key = |Ciphertext – Plaintext + 26| mod 26 = 9 which is J
- O (15) – C (3) Key = |Ciphertext – Plaintext + 26| mod 26 = 14 which is O
- M (13) – T (20) Key = |Ciphertext – Plaintext + 26| mod 26 = 7 which is H
- P (16) – C (3) Key = |Ciphertext – Plaintext + 26| mod 26 = 13 which is N

- Since Key is JOHN, decrypting LCTC5618{HCB_SXIUQ_CVL_OXL} with Vigenère using key JOHN gives us **COMP5618{YOU_FOUND_THE_BOX}**.

```
PS C:\Users\matin\AndroidStudioProjects\whatever>
PS C:\Users\matin\AndroidStudioProjects\whatever> cd "C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64"
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64> frida -U -f com.dozingcatsoftware.bouncy -l neww.js
     ____
    / _  |   Frida 17.4.0 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `com.dozingcatsoftware.bouncy`. Resuming main thread!
[Android Emulator 5554::com.dozingcatsoftware.bouncy ]-> [*] Keepalive: patched baseUrlFromJNI to a harmless URL.
[FLAG] COMP5618{JumBled_n_cOmpileD}
[*] Patched baseUrlFromJNI to return full flag.
```

```
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64>
PS C:\Users\matin\Downloads\Assignment 3 Files\challenge3\lib\x86_64> frida -U -f com.dozingcatsoftware.bouncy -l new.js
     ____
    / _  |   Frida 17.4.0 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `com.dozingcatsoftware.bouncy`. Resuming main thread!
[Android Emulator 5554::com.dozingcatsoftware.bouncy ]-> [JNI] baseUrlFromJNI -> COMP5618{
```