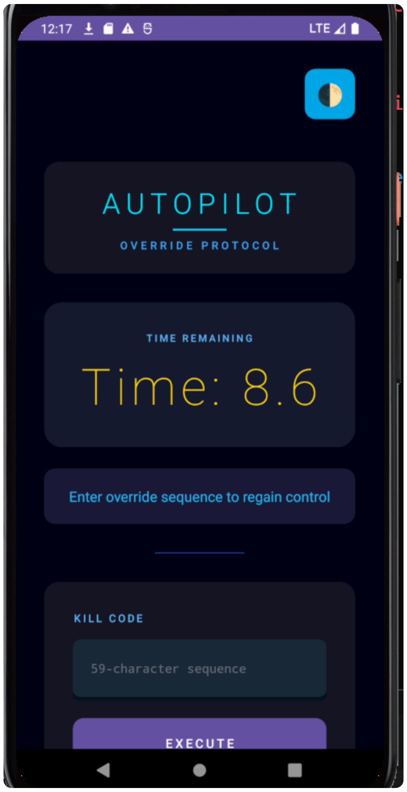
Cersei on autopilot write up

Author : yaseen

Description

```
Autopilot took full control of the plane and gone insane, figure out
what Cersei has to do with this and deactivate the autopilot to regain
control.
```

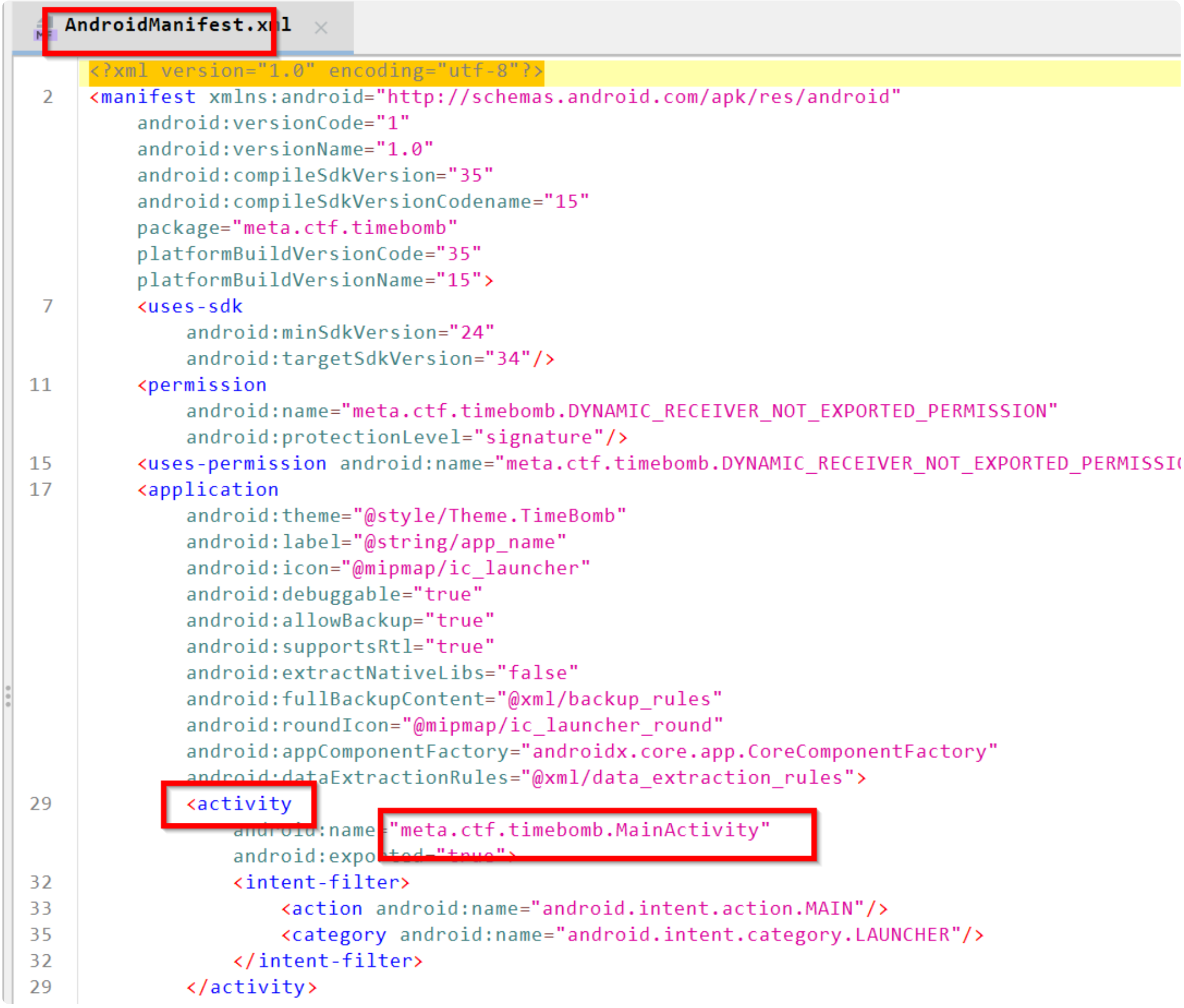
run on emulator



- We can tell it's asking for kill code, we can assume it's the flag
- We can see that it's 59 characters
- There is 10 seconds time limit so we can't guess the flag on the emulator simply

reversing

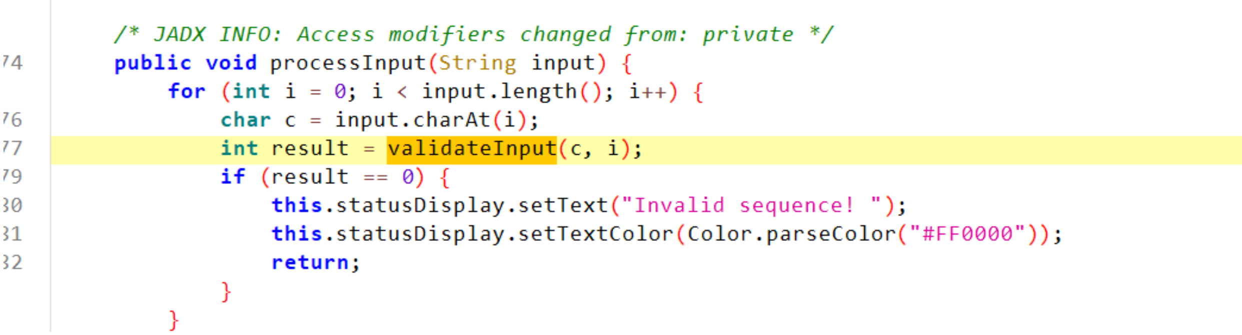
- using jadx to reverse we check the apk manifists to figure out the activity it does



AndroidManifest.xml

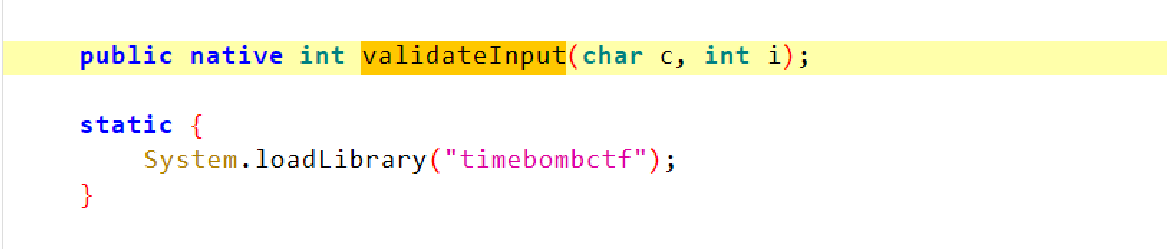
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    android:compileSdkVersion="35"
    android:compileSdkVersionCodename="15"
    package="meta.ctf.timebomb"
    platformBuildVersionCode="35"
    platformBuildVersionName="15">
    <uses-sdk
        android:minSdkVersion="24"
        android:targetSdkVersion="34"/>
    <permission
        android:name="meta.ctf.timebomb.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
        android:protectionLevel="signature"/>
    <uses-permission android:name="meta.ctf.timebomb.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSI
    <application
        android:theme="@style/Theme.TimeBomb"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:debuggable="true"
        android:allowBackup="true"
        android:supportsRtl="true"
        android:extractNativeLibs="false"
        android:fullBackupContent="@xml/backup_rules"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:appComponentFactory="androidx.core.app.CoreComponentFactory"
        android:dataExtractionRules="@xml/data_extraction_rules">
        <activity
            android:name="meta.ctf.timebomb.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

- Double click on the main activity to check it out
- Tracing the code we can see it calls validateInput() on our input



```
    /* JADX INFO: Access modifiers changed from: private */
    public void processInput(String input) {
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            int result = validateInput(c, i);
            if (result == 0) {
                this.statusDisplay.setText("Invalid sequence! ");
                this.statusDisplay.setTextColor(Color.parseColor("#FF0000"));
                return;
            }
        }
```

- Trying the check the validateInput function, we see it's not fully shown, instead it's calling a native-library



```
    public native int validateInput(char c, int i);

    static {
        System.loadLibrary("timebombctf");
    }
```

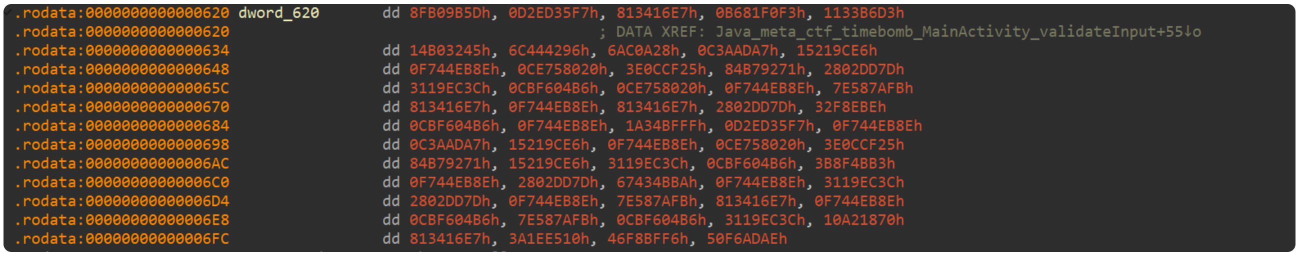
- Meaning the flag validation is happening inside the library and not directly in the java source code

Extracting the lib

1. either using `apktool -d `or by converting the extenstion of the apk to `.zip` then unzip
2. └───Cersei on autopilot - Copy
   ├───kotlin
   │ ├───annotation
   │ ├───collections
   │ ├───coroutines
   │ ├───internal
   │ ├───ranges
   │ └───reflect
   ├───lib
   │ ├───arm64-v8a
   │ ├───armeabi-v7a
   │ ├───x86
   │ └───x86_64
   3. all the libs are the same in functionality, choose what suits you to reverse it
   4. open it in ida, ghidra...

```
_BOOL8 __fastcall
Java_meta_ctf_timebomb_MainActivity_validateInput(__int64 a1, __int64
a2, char a3, unsigned int a4)
{
  return a4 < 59 && hash_byte(a3, 0x5Au) == dword_620[a4];
}
```

- Function Purpose: This is the main validation function called from Java via JNI (Java Native Interface).
- a3: The character being checked
- a4: Position in the string (0-58, max length 59)
- Computes hash_byte(character, 0x5A) and compares it to precomputed values in dword_620 array where 0x5A is the salt for hash_byte()



```
.rodata:0000000000000620 dword_620       dd 8FB09B5Dh, 0D2ED35F7h, 813416E7h, 0B681F0F3h, 1133B6D3h
.rodata:0000000000000620                                         ; DATA XREF: Java_meta_ctf_timebomb_MainActivity_validateInput+55↓o
.rodata:0000000000000634                 dd 14B03245h, 6C444296h, 6AC0A28h, 0C3AADA7h, 15219CE6h
.rodata:0000000000000648                 dd 0F744EB8Eh, 0CE758020h, 3E0CCF25h, 84B79271h, 2802DD7Dh
.rodata:000000000000065C                 dd 3119EC3Ch, 0CBF604B6h, 0CE758020h, 0F744EB8Eh, 7E587AFBh
.rodata:0000000000000670                 dd 813416E7h, 0F744EB8Eh, 813416E7h, 2802DD7Dh, 32F8EBEh
.rodata:0000000000000684                 dd 0CBF604B6h, 0F744EB8Eh, 1A34BFFFh, 0D2ED35F7h, 0F744EB8Eh
.rodata:0000000000000698                 dd 0C3AADA7h, 15219CE6h, 0F744EB8Eh, 0CE758020h, 3E0CCF25h
.rodata:00000000000006AC                 dd 84B79271h, 15219CE6h, 3119EC3Ch, 0CBF604B6h, 3B8F4BB3h
.rodata:00000000000006C0                 dd 0F744EB8Eh, 2802DD7Dh, 67434BBAh, 0F744EB8Eh, 3119EC3Ch
.rodata:00000000000006D4                 dd 2802DD7Dh, 0F744EB8Eh, 7E587AFBh, 813416E7h, 0F744EB8Eh
.rodata:00000000000006E8                 dd 0CBF604B6h, 7E587AFBh, 0CBF604B6h, 3119EC3Ch, 10A21870h
.rodata:00000000000006FC                 dd 813416E7h, 3A1EE510h, 46F8BFF6h, 50F6ADAEh
```

- It contains 59 values, each value is 4-bytes representing *a hashed byte* from input

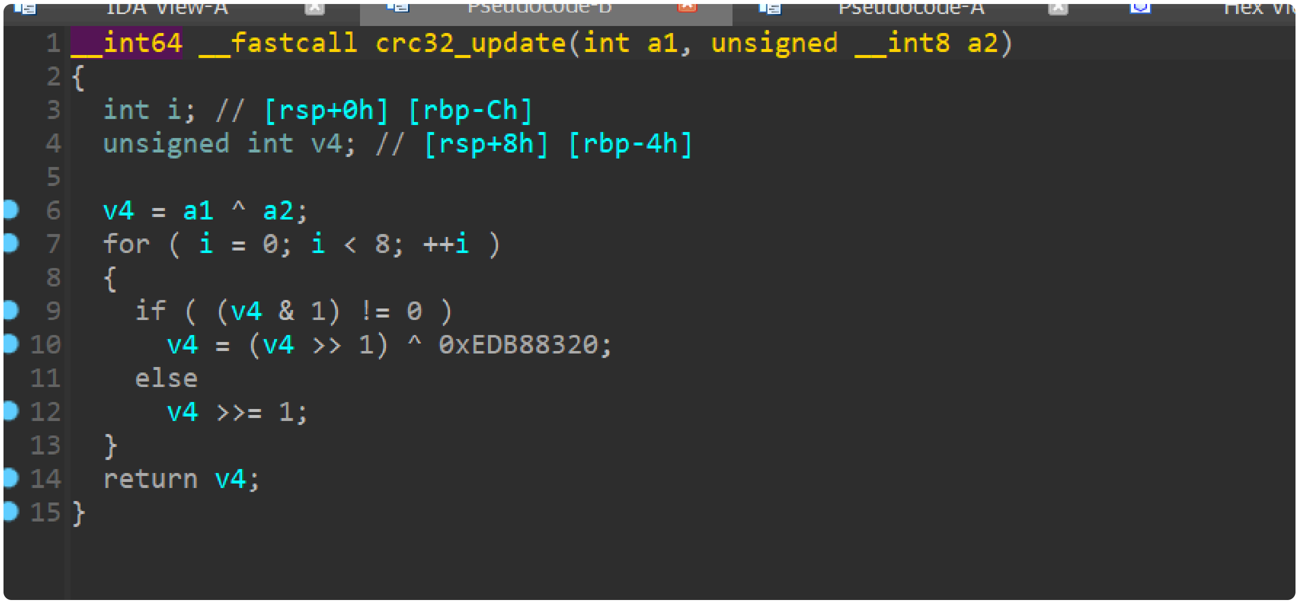
Hash Function

```
__int64 __fastcall hash_byte(unsigned __int8 a1, unsigned __int8 a2)
{
  unsigned int v3; // [rsp+8h] [rbp-8h]

  v3 = crc32_update(0xFFFFFFFF, a1);
  return (unsigned int)~crc32_update(v3, a2);
}
```

- Two-stage CRC32 hash
- CRC32 the input character with initial value 0xFFFFFFFF
- CRC32 the result with salt 0x5A
- Return bitwise NOT of final result

CRC32 Implementation



```
__int64 __fastcall crc32_update(int a1, unsigned __int8 a2)
{
  int i; // [rsp+0h] [rbp-Ch]
  unsigned int v4; // [rsp+8h] [rbp-4h]

  v4 = a1 ^ a2;
  for ( i = 0; i < 8; ++i )
  {
    if ( (v4 & 1) != 0 )
      v4 = (v4 >> 1) ^ 0xEDB88320;
    else
      v4 >>= 1;
  }
  return v4;
}
```

- Standard CRC32 algorithm processing one byte
- XOR input with current state
- Process each bit with 0xEDB88320

conclusion

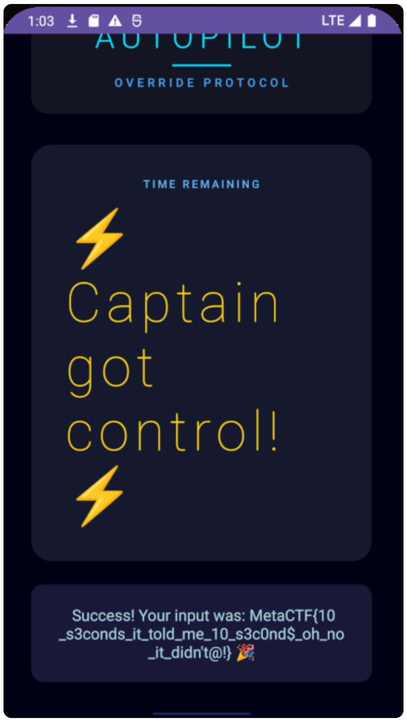
Since we know:

1. The hash function
2. The target hash values (dword_620 array)
3. Characters are likely printable ASCII
   We can brute force each position:

```
def brute_force_flag():
    flag_chars = []
    for i in range(len(dword_620)):
        for c in string.printable:
            if hash_byte(ord(c), 0x5A) == dword_620[i]:
                flag_chars.append(c)
                break
    return "".join(flag_chars)
```

another way of solving would be using frida-instrument and checking when does the validateInput() function returns 1

check the solve.py



AUTOPILOT

OVERRIDE PROTOCOL

TIME REMAINING

⚡ Captain got control! ⚡

Success! Your input was: MetaCTF{10_s3conds_it_told_me_10_s3c0nd$_oh_no_it_didn't@!} 🎉