

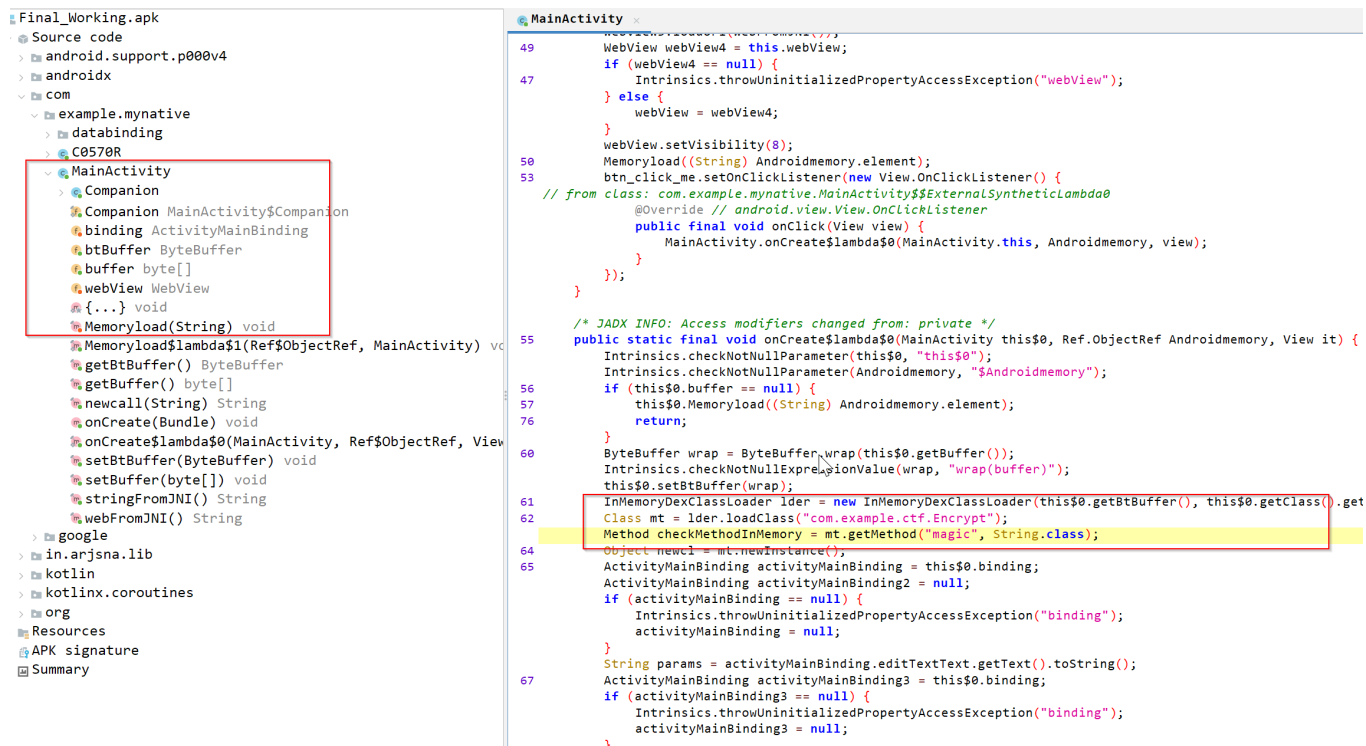
Solution for MemDump

Memdump CTF Solution (arm64-v8a)



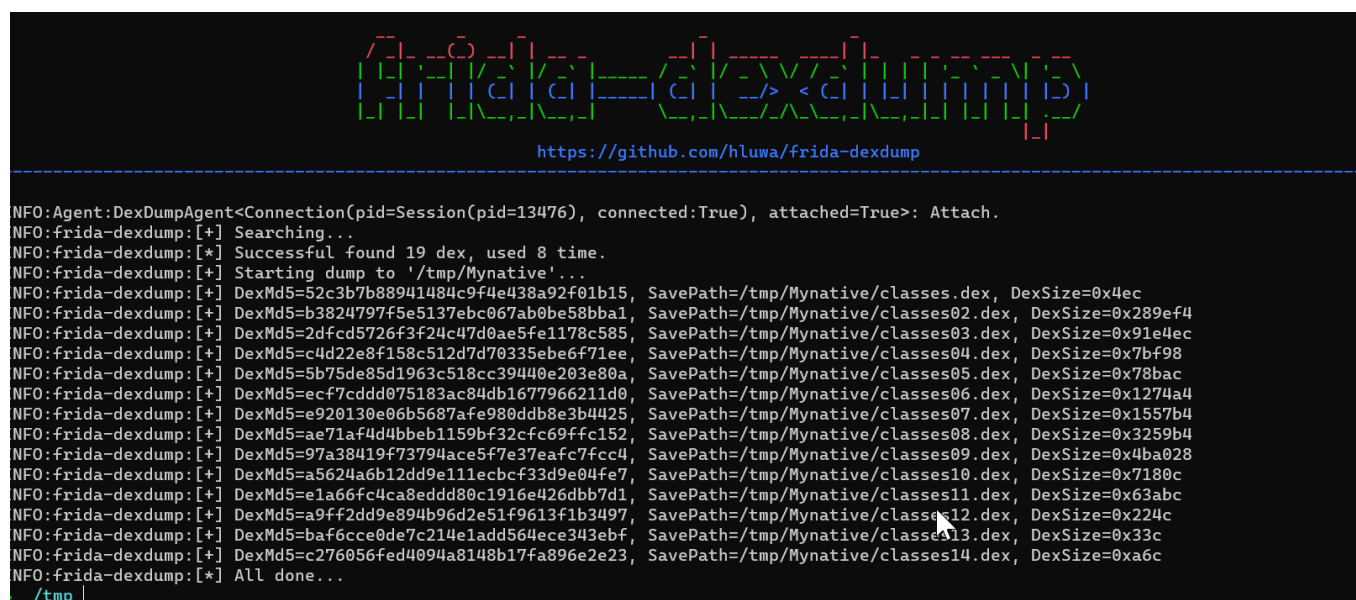
Using jadx check the apk source code

By analyzing the source code we can observe that magic method is getting loaded from "InmemoryDexloader" it means the dex file is stored in application memory

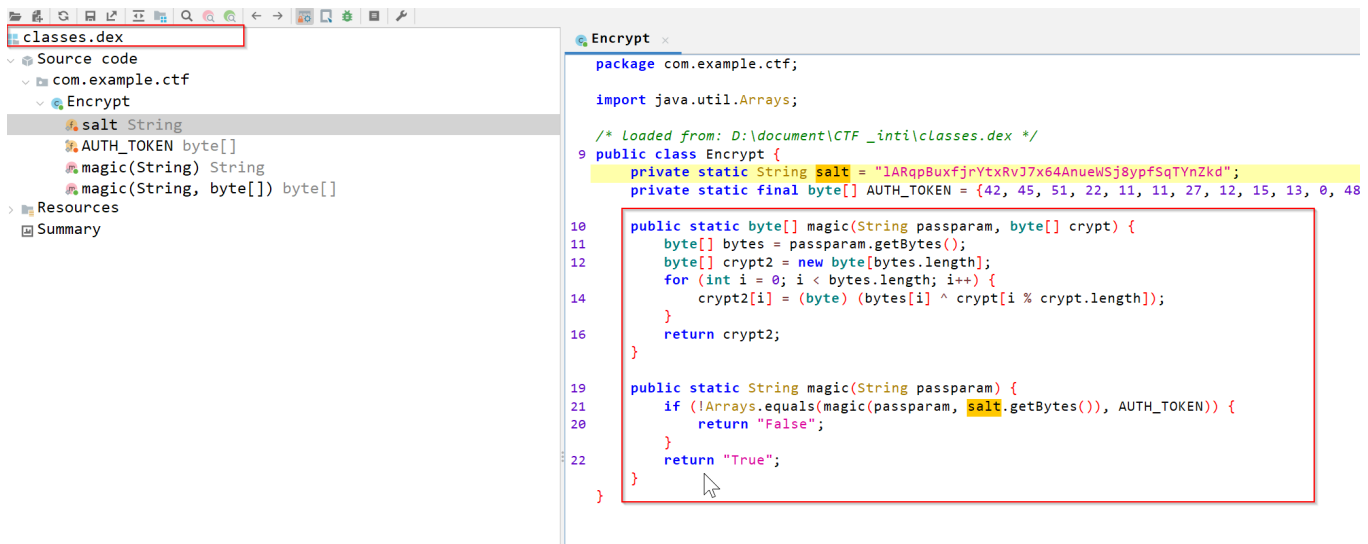


Dumping the dex file using frida

<https://github.com/hluwa/frida-dexdump>



Open the classes.dex file in jadx and observed that magic function is there and it perform Bytecode XOR of salt and AUTH_TOKEN to get flag



By writing the below java code we can obtain the flag

```

import java.nio.charset.StandardCharsets;

public class Main
{
    public static void main(String[] args) {

        String salt = "lARqpBuxfjrYtxRvJ7x64AnueWSj8ypfSqTYnZkd";

        final byte[] AUTH_TOKEN={42, 45, 51, 22, 11, 11, 27, 12, 15, 13, 0, 48,
0, 17, 33, 6, 47, 84, 17, 87, 88};

        byte[] crypt = salt.getBytes();
        byte[] params = new byte[AUTH_TOKEN.length];
        for (int i = 0; i < AUTH_TOKEN.length; i++) {
            params[i] = (byte) (AUTH_TOKEN[i] ^ crypt[i % crypt.length]);
        }

        System.out.printf("%s", new String(params));

    }
}

```

First part of flag

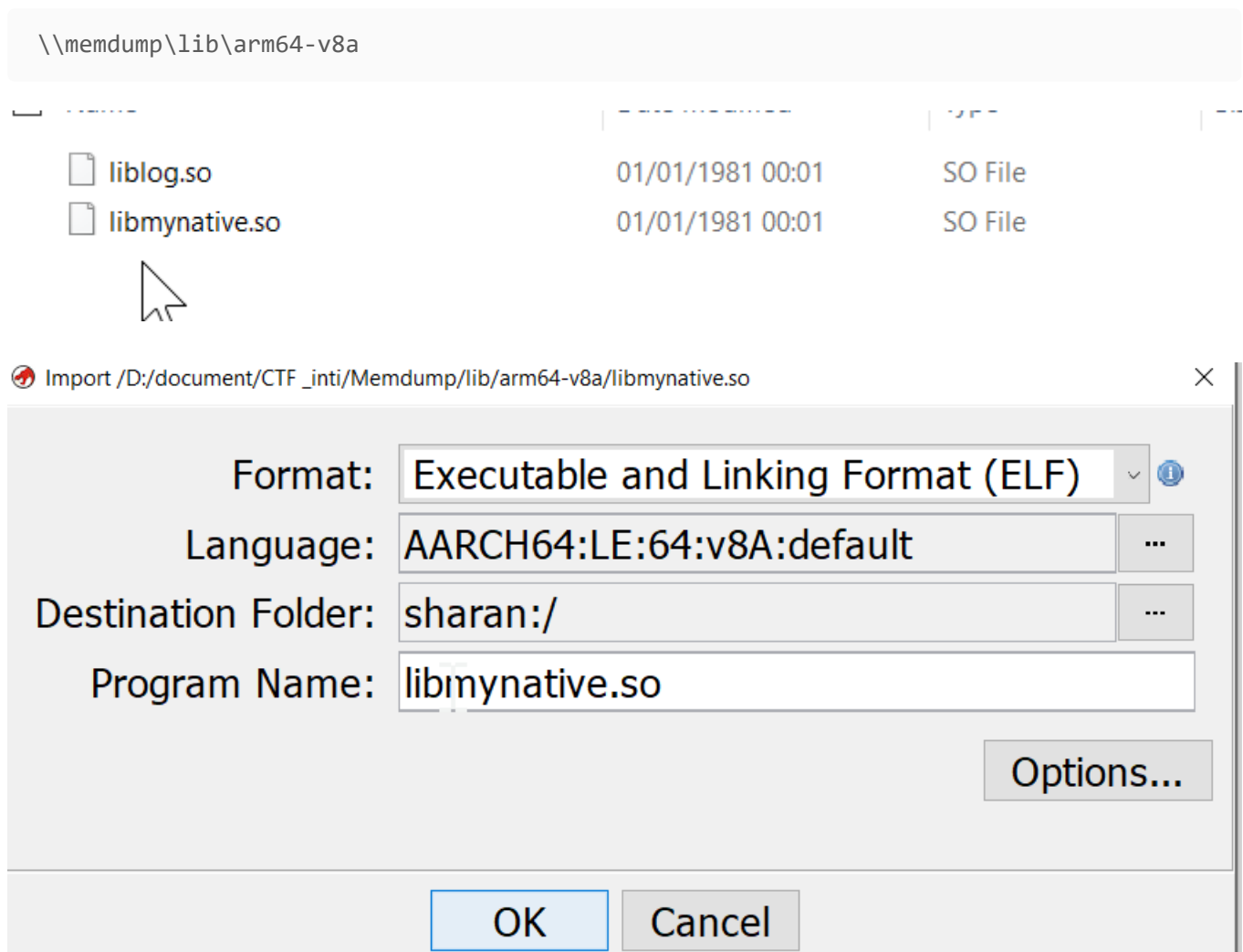
Flag{Intigritispecial

Second part of flag

By again observing with second edittext we can observe the newcall is getting called which native function defined at the start

```
}
textView.setText(this$0.newcall(activityMainBinding5.editTextText2.getText().toString()));
if (Intrinsics.areEqual(checkMethodInMemory.invoke(newcl, params).toString(), "True")) {
    ActivityMainBinding activityMainBinding6 = this$0.binding;
    if (activityMainBinding6 == null) {
        Intrinsics.throwUninitializedPropertyAccessException("binding");
    } else {
        activityMainBinding2 = activityMainBinding6;
    }
    if (Intrinsics.areEqual(this$0.newcall(activityMainBinding2.editTextText2.getText().toString()), "True")) {
        Toast.makeText(this$0.getBaseContext(), "You solved it", 1).show();
    }
}
}
```

Native Code analysis using Ghidra



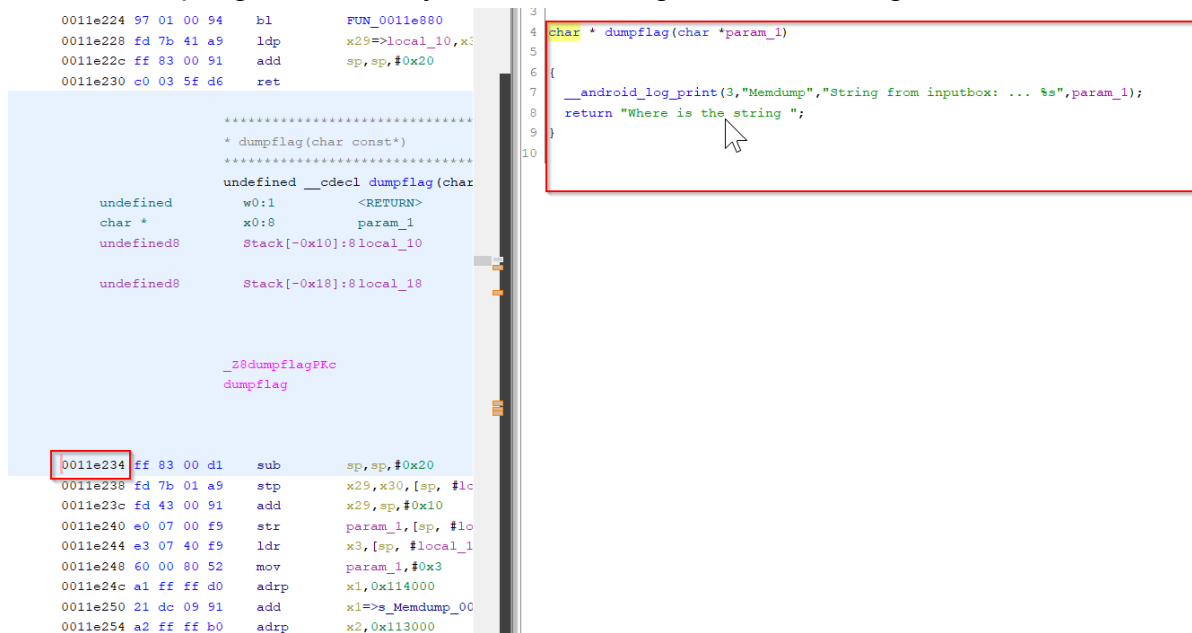
dumpflag function is getting called in native which return the variable which is get compared with if return value matches "RheO5PB6mfl5N3YBH45e5XuCEaWpvWUFESqTYnZk" flag will be leaked in logcat `__android_log_print()`

```

2 undefined8
3 Java_com_example_mynative_MainActivity_newcall
4     (_JNIEnv *param_1,undefined8 param_2,_jstring *param_3)
5
6 {
7     int iVar1;
8     long lVar2;
9     int iVar3;
10    char *pcVar4;
11    int local_2d4;
12    undefined8 local_298;
13    byte abStack576 [256];
14    byte abStack320 [256];
15    undefined8 uStack64;
16    undefined8 uStack56;
17    long local_28;
18
19    lVar2 = cRead_8(tpidr_e10);
20    local_28 = *(long *) (lVar2 + 0x28);
21    pcVar4 = (char *) JNIEnv::GetStringUTFChars(param_1,param_3,(uchar *)0x0);
22    pcVar4 = (char *) dumpflag(pcVar4);
23    if (pcVar4 == "RheO5PB6mFL5N3YBH45e5XuCEaWpvWUFESqTYnZk") {
24        uStack56 = 0x3f2a432354240505;
25        uStack64 = 0x422b274624060720;
26        iVar3 = __strlen_chk("RheO5PB6mFL5N3YBH45e5XuCEaWpvWUFESqTYnZk",0xffffffffffffffff);
27        __memcpy_chk(abStack320,"RheO5PB6mFL5N3YBH45e5XuCEaWpvWUFESqTYnZk",(long)iVar3,0x100);
28        __memset_chk(abStack576,0,0x10,0x100);
29        for (local_2d4 = 0; local_2d4 < 0x10; local_2d4 = local_2d4 + 1) {
30            iVar1 = 0;
31            if (iVar3 != 0) {
32                iVar1 = local_2d4 / iVar3;
33            }
34            abStack576[local_2d4] =
35                *(byte *) ((long)&uStack64 + (long)local_2d4 ^ abStack320[local_2d4 - iVar1 * iVar3]);
36        }
37        __android_log_print(3,"Memdump","Success Second part: ... %s",abStack576);
38        local_298 = JNIEnv::NewStringUTF(param_1,"True");
39    }
40    else {
41        __android_log_print(3,"Memdump","Wrong value: ... %s",pcVar4);
42        local_298 = JNIEnv::NewStringUTF(param_1,"False");
43    }
44    lVar2 = cRead_8(tpidr_e10);
45    if (*(long *) (lVar2 + 0x28) == local_28) {
46        return local_298;
47    }
48
49    /* WARNING: Subroutine does not return */
50    __stack_chk_fail();
51 }

```

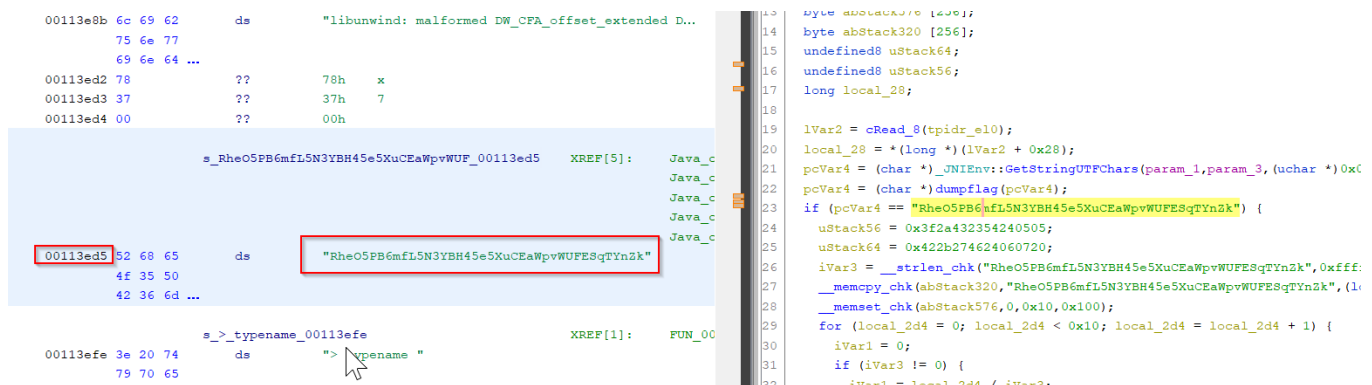
For the dumpflag function only return the string "Where is string "



The screenshot shows the Ghidra interface with assembly and decompiled code for the `dumpflag` function. The assembly view on the left shows instructions starting at address `0011e234`. The decompiled code on the right shows the function signature `char * dumpflag(char *param_1)` and the return statement `return "Where is the string ";`. A red box highlights the return statement in the decompiled code.

Note the memory address of dumpflag function "0x0011e234"

Now to return the string "RheO5PB6mfl5N3YBH45e5XuCEaWpWUFESqTYnZk" using frida we need find the address of the above string which is "0x00113ed5"



The screenshot shows the Ghidra interface with assembly and decompiled code for the `dumpflag` function. The assembly view on the left shows instructions starting at address `00113ed5`. The decompiled code on the right shows the function signature `char * dumpflag(char *param_1)` and the return statement `return "RheO5PB6mfl5N3YBH45e5XuCEaWpWUFESqTYnZk";`. A red box highlights the return statement in the decompiled code.

We have

dumpflag = 0x0011e234

string = 0x00113ed5

using below frida script we can return the value of dumpflag to pointer 0x00113ed5

```
const ghidraImageBase = 0x00100000; // example value get the real value in Ghidra
from Window -> Memory map -> Set Image Base
const moduleName = "libmynative.so";
const moduleBaseAddress = Module.findBaseAddress(moduleName);
const functionRealAddress = moduleBaseAddress.add(0x0011e234 - ghidraImageBase);
const functionRealAddress2 = moduleBaseAddress.add(0x00113ed5 - ghidraImageBase);
Interceptor.attach(functionRealAddress, {
  onEnter: function(args) {
```

```

        return 0;

    },
    onLeave: function(retval) {

        console.log(Memory.readCString(functionRealAddress2))
        retval.replace(functionRealAddress2)
        console.log("Flag Found")

    },

});

```

```

const moduleName = "libmynative.so";
const moduleBaseAddress = Module.findBaseAddress(moduleName);
const functionRealAddress = moduleBaseAddress.add(0x0011e234 - ghidraImageBase);
const functionRealAddress2 = moduleBaseAddress.add(0x00113ed5 - ghidraImageBase);
Interceptor.attach(functionRealAddress, {
    onEnter: function(args) {
        return 0;
    },
    onLeave: function(retval) {

        console.log(Memory.readCString(functionRealAddress2))
        retval.replace(functionRealAddress2)
        console.log("Flag Found")

    },
});
[GM1913::com.example.mynative ]-> Rhe05PB6mfL5N3YBH45e5XuCEaWpvWUFESqTYnZk
Flag Found

```

```

07-20 00:23:42.936 19000 19108 W cr_VAUtil: Unknown profile: 2 or level: 131072 for c
07-20 00:23:42.936 19000 19108 W cr_VAUtil: Unknown profile: 524288 or level: 131072
07-20 00:23:42.936 19000 19108 W cr_VAUtil: Unknown profile: 8 or level: 131072 for c
07-20 00:23:47.539 19000 19071 I cr_X509Util: Failed to validate the certificate chain, error: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
07-20 00:23:47.540 19000 19098 E chromium: [ERROR:ssl_client_socket_impl.cc(978)] handshake failed; returned -1, SSL error code 1, net_error -202
07-20 00:23:48.051 19000 19131 E seccheck: -1

07-20 00:23:57.909 19000 19194 D seccheck: Success of download to buffer

07-20 00:24:11.698 19000 19000 I AssistStructure: Flattened final assist data: 2064 bytes, containing 1 windows, 12 views
07-20 00:24:11.716 19000 19000 E SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
07-20 00:24:11.720 19000 19000 I chatty : uid=10311(com.example.mynative) identical 2 lines
07-20 00:24:11.720 19000 19000 E SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length
07-20 00:24:15.425 19000 19000 D Memdump : String from inputbox: ...
07-20 00:24:15.425 19000 19000 D Memdump : Success Second part: ... rockswithchamps}

```

Second part of flag

rockswithchamps}

Full flag

Flag{Intigritispecialrockswithchamps}