

# BÀI TẬP CTF

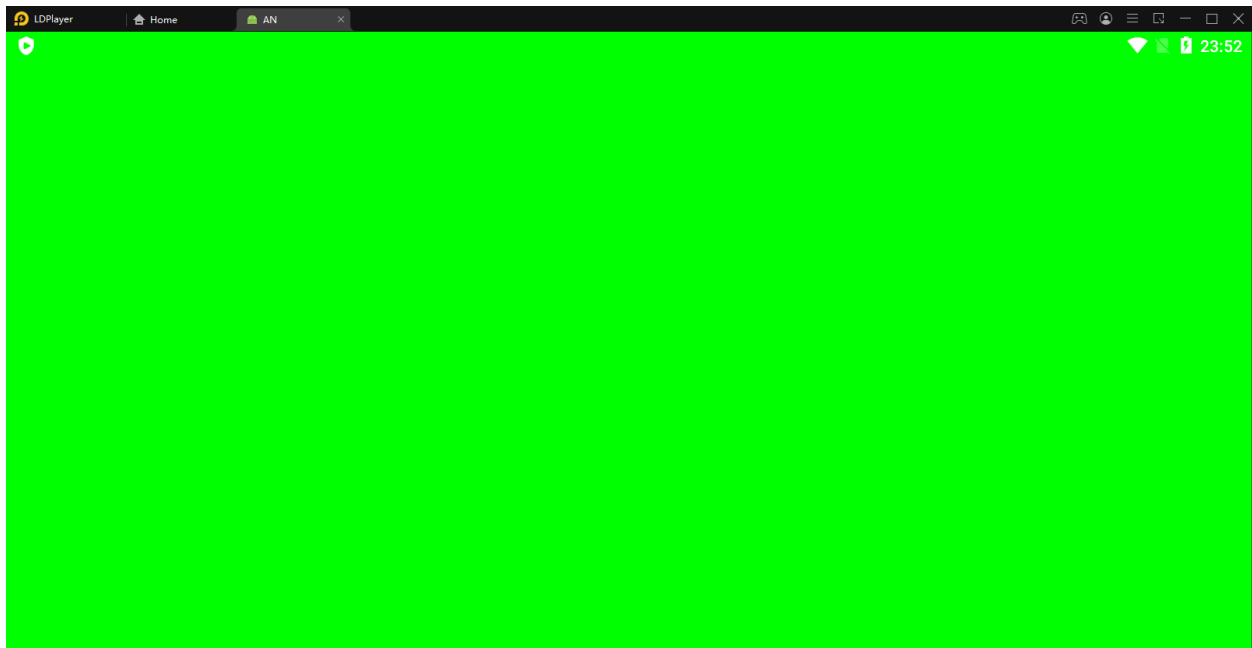
## Bảo mật web và ứng dụng – NT213.M21.ANTN

**Giảng viên hướng dẫn:** *Đỗ Hoàng Hiển*

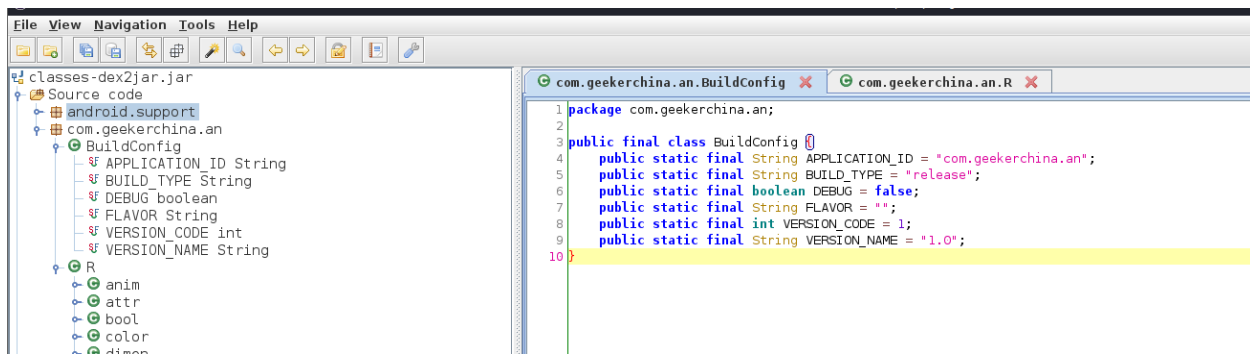
**Sinh viên thực hiện:** *19520199 – Lê Tôn Nhân*

### Challenge 2: AN

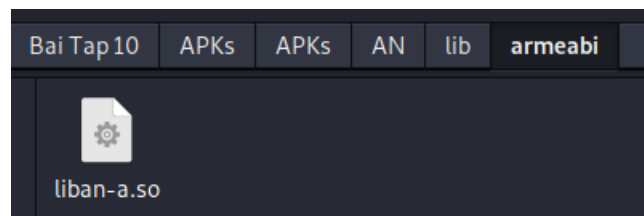
- Cài đặt ứng dụng, ở đây em sử dụng LDPlayer để chạy ứng dụng
- Ta thấy ứng dụng chỉ gồm một màu xanh và không có gì



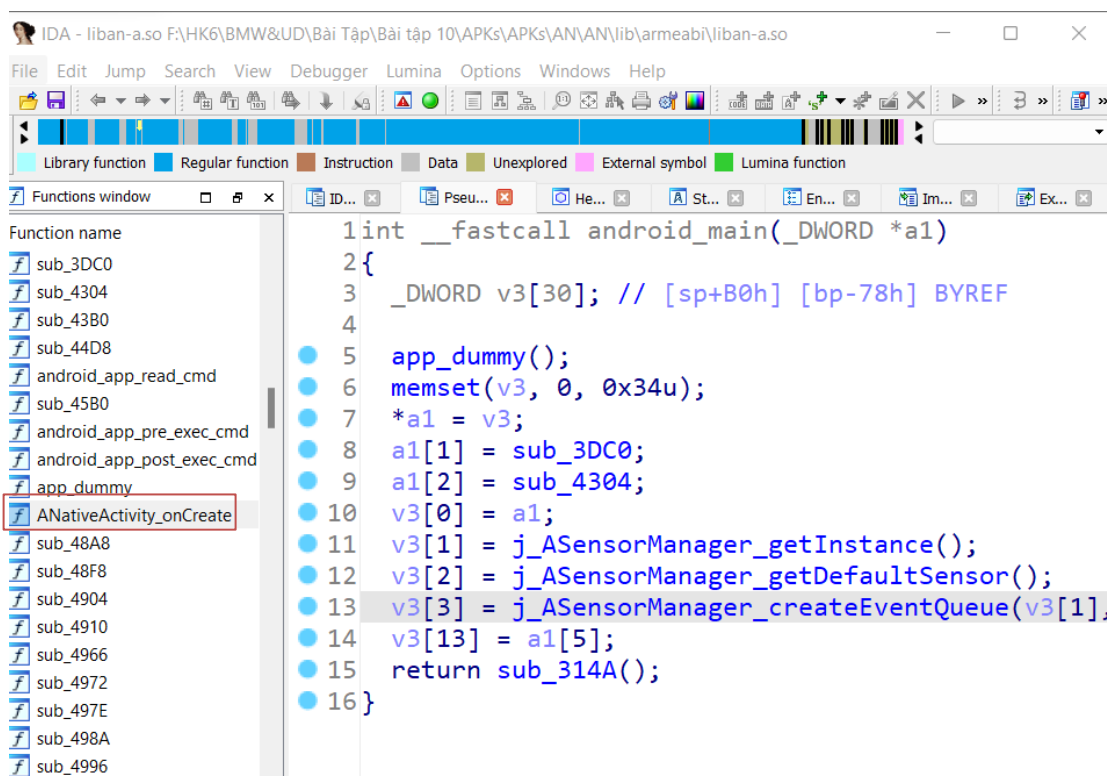
- Giải nén file AN.apk, sau đó sử dụng dex2jar để chuyển file class.dex thành file jar
- Sử dụng jadx-gui để dịch ngược ứng dụng và xem các logic cơ bản của nó
- Ta thấy không có thông tin gì cả có thể chương trình đã được load từ thư viện .so lên và chạy trực tiếp từ hàm onload hoặc oncreate



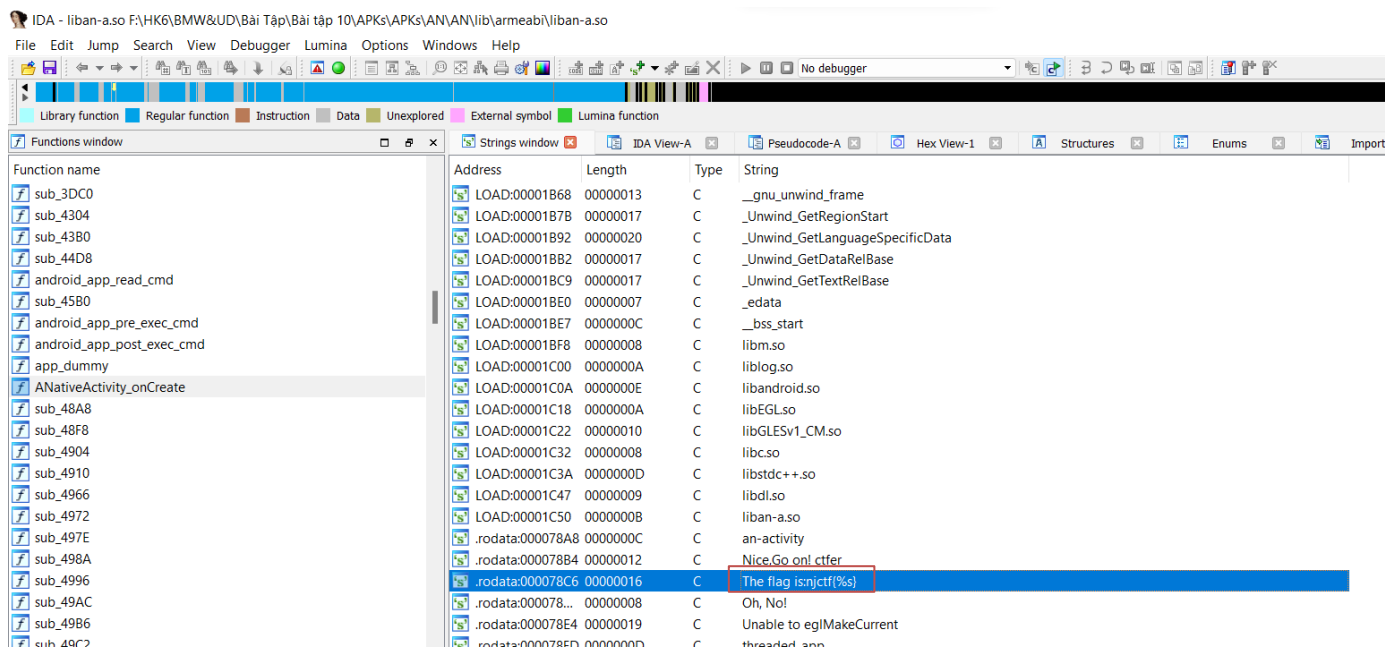
- Sau khi decompiler file APK ta có file **liban-a.so**



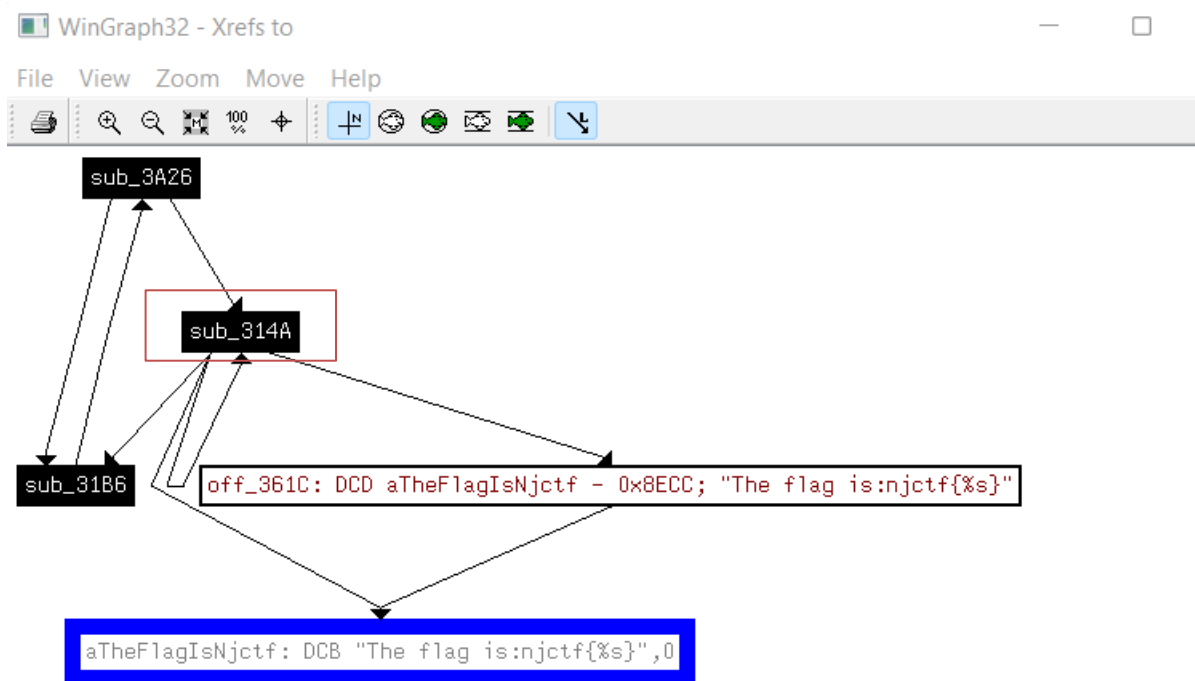
- Kiểm tra file này với IDA, thì thấy có các hàm **oncreate** giống như ta dự đoán



- Sử dụng của sổ strings của IDA ta tìm thấy chuỗi liên quan đến flag



- Sử dụng xrefs to trong IDA để tìm các hàm liên quan đến chuỗi flag

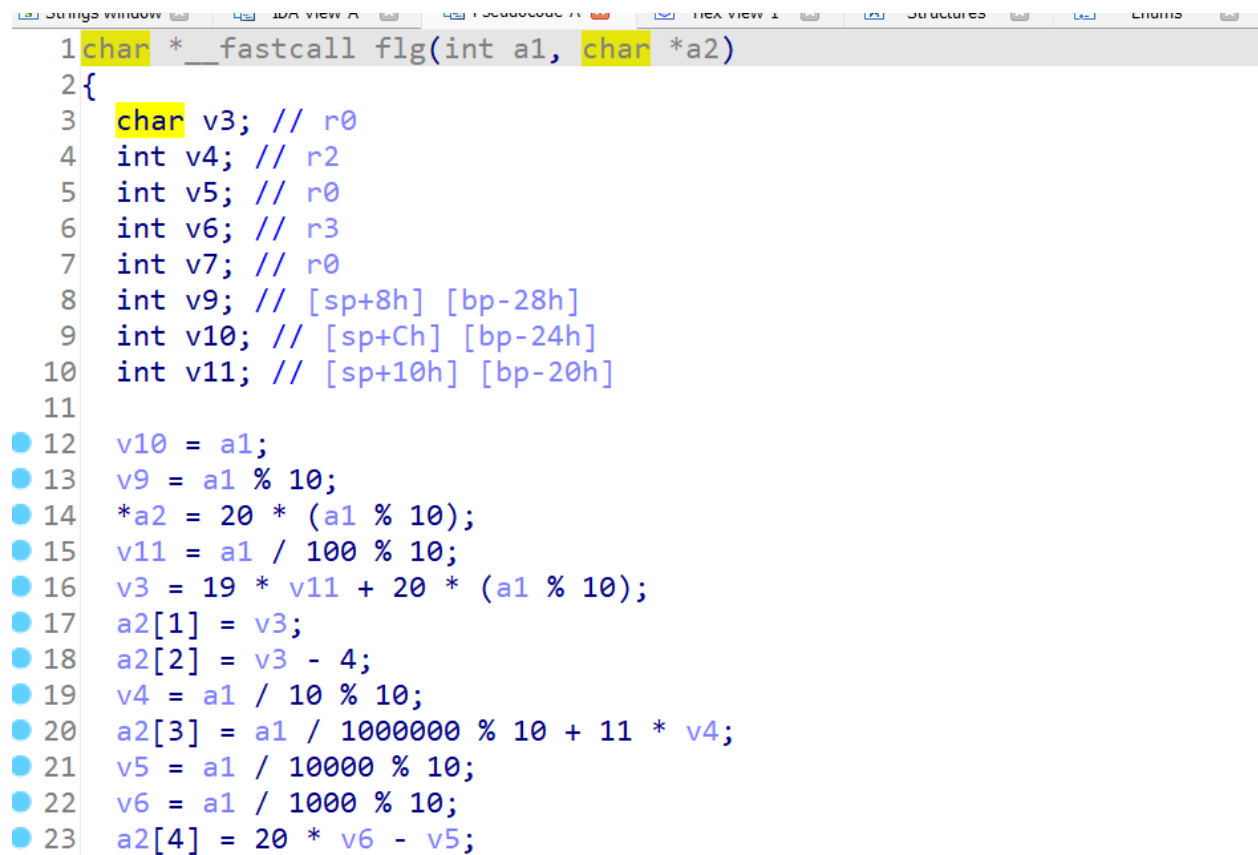


- Ta thấy được hàm **sub\_314A** có liên quan đến chuỗi nào. Vào hàm này kiểm tra. Ở trong hàm này ta thấy gọi tới hàm flg và lưu vào biến **v18**.

Sau đó in ra dòng **The falg is:njctf{%s} v18**. Do đó biến **v18** chính là flag mà ta cần tìm.

```
v18 = (const char *)flg(v31, v42);
j__android_log_print(
    4,
    "an-activity",
    "The flag is:njctf{%s}",
    v18);
```

- Giờ ta đi phân tích hàm flg



```
1 char * __fastcall flg(int a1, char *a2)
2 {
3     char v3; // r0
4     int v4; // r2
5     int v5; // r0
6     int v6; // r3
7     int v7; // r0
8     int v9; // [sp+8h] [bp-28h]
9     int v10; // [sp+Ch] [bp-24h]
10    int v11; // [sp+10h] [bp-20h]
11
12    v10 = a1;
13    v9 = a1 % 10;
14    *a2 = 20 * (a1 % 10);
15    v11 = a1 / 100 % 10;
16    v3 = 19 * v11 + 20 * (a1 % 10);
17    a2[1] = v3;
18    a2[2] = v3 - 4;
19    v4 = a1 / 10 % 10;
20    a2[3] = a1 / 1000000 % 10 + 11 * v4;
21    v5 = a1 / 10000 % 10;
22    v6 = a1 / 1000 % 10;
23    a2[4] = 20 * v6 - v5;
```

```

23 a2[4] = 20 * v6 - v5;
24 a2[5] = (v4 + v9) * v6;
25 a2[6] = v4 * v6 * v5;
26 v7 = v10 / 100000 % 10;
27 a2[7] = 20 * v7 - v11;
28 a2[8] = (10 * v6) | 1;
29 a2[9] = (v4 + v9) * v7 - 1;
30 a2[10] = v9 * v4 * v11 * v11 - 4;
31 *(_WORD *) (a2 + 11) = (unsigned __int8)((v11 + v4) * v7 - 5);
32 return a2;
33 }

```

- Ta biết kết quả trả về của hàm này chính là flag do đó ta lấy nguyên hàm này và sử dụng lại. Sau đó brute force để tìm ra các giá trị a1 tương ứng với các từ của flag
- Đoạn mã brute force

```

(kali@kali)~[~/Bai Tap 10/APKs/APKs/AN]
$ cat an.c
#include<stdio.h>
int j___modsi3(int a,int b)
{
    return a%b;
}
int j___divsi3(int a,int b)
{
    return a/b;
}
char flg(int a1, char *out)
{
    char *v2; // r6@1
    int v3; // ST0C_4@1
    int v4; // r4@1
    int v5; // r0@1
    int v6; // ST08_4@1
    int v7; // r5@1
    int v8; // r0@1
    int v9; // r0@1
    char v10; // ST10_1@1
    int v11; // r0@1
    int v12; // r5@1
    int v13; // r0@1
    int v14; // ST18_4@1
    int v15; // r0@1
    int v16; // r0@1
    char v17; // r0@1
    char v18; // ST04_1@1
    int v19; // r0@1
    char v20; // r0@1
    int v21; // r1@1
    int v22; // r5@1
    int v23; // r0@1
    char v24; // r0@1
    v2 = out;
    v3 = a1;
    v4 = a1;
    v5 = j___modsi3(a1, 10);
    v6 = v5;
    v7 = 20 * v5;
    *v2 = 20 * v5;
    v8 = j___divsi3(v4, 100);
    v9 = j___modsi3(v8, 10);
}
package com.geekerchina.an;

public final class BuildConfig {
    public static final String
    public static final String
    public static final boolean
    public static final String
    public static final int VER
    public static final String
}

```

```
The Actions Edit View Help
v2[3] = j_ j___modsi3(v15, 10) + 11 * v14;
v16 = j_ j___divsi3(v4, 1000);
v17 = j_ j___modsi3(v16, 10);
//LOBYTE(v4) = v17;
v4 = v17;
v18 = v17;
v19 = j_ j___divsi3(v12, 10000);
v20 = j_ j___modsi3(v19, 10);
v2[4] = 20 * v4 + 60 - v20 - 60;
v21 = -v6 - v14;
v22 = -v21;
v2[5] = -(char)v21 * v4;
v2[6] = v14 * v4 * v20;
v23 = j_ j___divsi3(v3, 100000);
v24 = j_ j___modsi3(v23, 10);
v2[7] = 20 * v24 - v10;
v2[8] = 10 * v18 | 1;
v2[9] = v22 * v24 - 1;
v2[10] = v6 * v14 * v10 * v10 - 4;
v2[11] = (v10 + v14) * v24 - 5;
v2[12] = 0;
return v2;
}
int main()
{
    char out[256], flag = 0;
    for(unsigned int i=0; i<=4294967295-1; ++i)
    {
        flag = 0;
        memset(out, 0, 256);
        flg(i, out);
        if(strlen(out) >= 10)
        {
            for(int j=0; j<12; ++j)
            {
                if((out[j] >= 'a' && out[j] <= 'z') || (out[j] >= 'A' && out[j] <= 'Z') || (out[j] >= '0' && out[j] <= '9') || out[j] == '_')
                    continue;
                else
                {
                    flag = 1;
                    break;
                }
            }
            if(flag == 0)
                printf("%sn\n", out);
        }
    }
    return 0;
}
```

- Kết quả thu được

```
(kali㉿kali)-[~/.../Bai Tap 10/APKs/APKs/AN]
$ chmod 777 *
BUILD TYPE String
FLAVOR String
PvrMb7Fv3Al1n VERSION_CODE int
PvrMa7iv3Al1n VERSION_NAME String
PvrNb7Fv3Al1n
PvrNa7iv3Al1n
PvrOb7Fv3Al1n
PvrOa7iv3Al1n
PvrPb7Fv3Al1n
PvrPa7iv3Al1n
PvrQb7Fv3Al1n
PvrQa7iv3Al1n
PvrRb7Fv3Al1n
PvrRa7iv3Al1n
PvrSb7Fv3Al1n
PvrSa7iv3Al1n
PvrTb7Fv3Al1n
PvrTa7iv3Al1n
PvrUb7Fv3Al1n
PvrUa7iv3Al1n
PvrVb7Fv3Al1n
PvrVa7iv3Al1n
PvrMb7Fv3Al1n
PvrMa7iv3Al1n
PvrNb7Fv3Al1n
PvrNa7iv3Al1n
PvrOb7Fv3Al1n
PvrOa7iv3Al1n
PvrPb7Fv3Al1n
PvrPa7iv3Al1n
PvrQb7Fv3Al1n
PvrQa7iv3Al1n
PvrRb7Fv3Al1n
```

**Flag:njctf{PvrNa7iv3Al1n}**