BÀI TẬP CTF

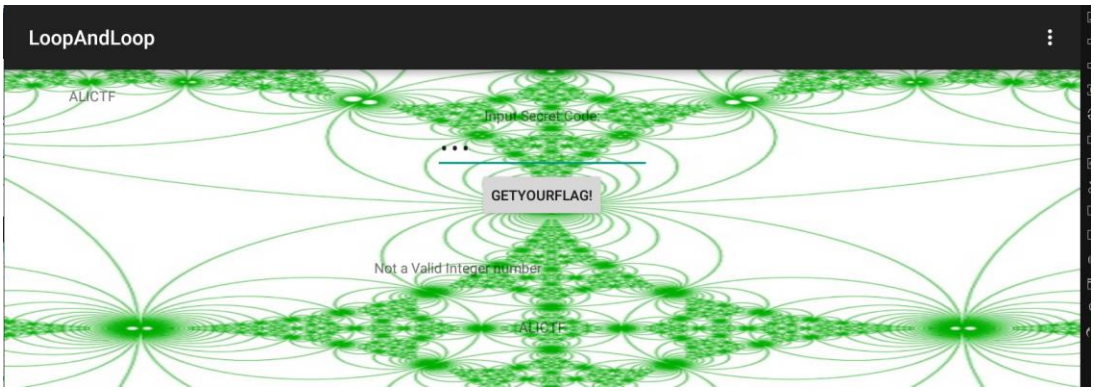
Bảo mật web và ứng dụng – NT213.M21.ANTN

Giảng viên hướng dẫn: *Đỗ Hoàng Hiển*

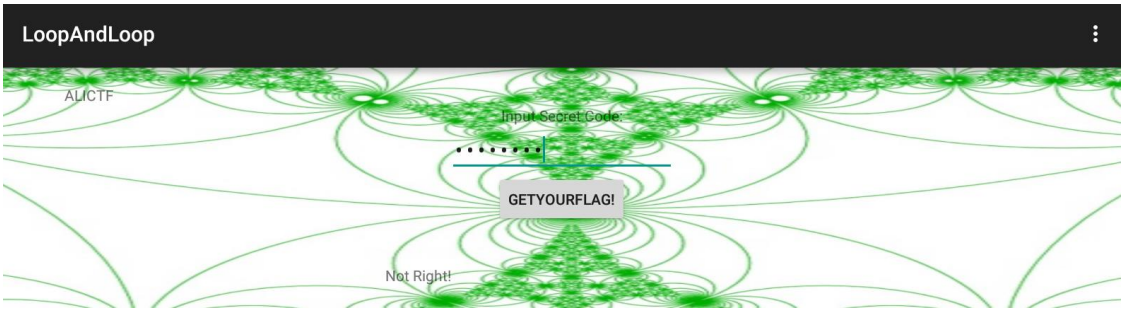
Sinh viên thực hiện: *19520199 – Lê Tôn Nhân*

Challenge 3: LoopAndLoop

- Cài đặt ứng dụng, ở đây em sử dụng LDPlayer để chạy ứng dụng
- Ta thấy ứng dụng yêu cầu ta nhập vào chuỗi secret code
- Thử nhập với chuỗi abc thì ta thấy kết quả xuất ra thông báo **Not a valid interger number**. Do đó chuỗi secret code là một dãy số



- Thử nhập với chuỗi 19520199 thì ta thấy dòng thông báo **Not right**



- Tiến hành giải nén file **loopandloop.apk** và sử dụng **dex2jar** để chuyển tập tin **class.dex** sang dạng jar
- Sau đó sử dụng **jadx-gui** để dịch ngược ứng dụng và xem các logic cơ bản của nó
- Phân tích hàm **onclick** ở **mainactivity**

```
public void onClick(View view) {
    try {
        int parseInt = Integer.parseInt(editText.getText().toString());
        if (MainActivity.this.check(parseInt, 99) == 1835996258) {
            textView.setText("The flag is:");
            textView2.setText("alictf{" + MainActivity.this.stringFromJNI2(parseInt) + "}");
            return;
        }
        textView.setText("Not Right!");
    } catch (NumberFormatException e) {
        textView.setText("Not a Valid Integer number");
    }
}
```

- Đầu tiên ta thấy lệnh **try-catch** với **NumberFormatException**. Đây là một ngoại lệ ở định dạng số, khi ta cố gắng chuyển đổi chuỗi sang kiểu số được chỉ định. Khi chuỗi không đáp ứng định dạng theo yêu cầu của loại số, thì sẽ sinh ra ngoại lệ. Do đó nếu ta nhập các ký tự không phải là số thì chương trình sẽ bắt ngoại lệ và xuất ra dòng "**Not a Valid Interger Number**"

```
public void onClick(View view) {
    try {
        int parseInt = Integer.parseInt(editText.getText().toString());
        if (MainActivity.this.check(parseInt, 99) == 1835996258) {
            textView.setText("The flag is:");
            textView2.setText("alictf{" + MainActivity.this.stringFromJNI2(parseInt) + "}");
            return;
        }
        textView.setText("Not Right!");
    } catch (NumberFormatException e) {
        textView.setText("Not a Valid Integer number");
    }
}
```

- Tiếp theo ta thấy biến **parseInt** sẽ lưu chuỗi đầu vào của chúng ta và chuyển đổi thành dạng số
- Tiếp theo là câu lệnh **if-else**. Ở lệnh này sẽ kiểm tra xem điều kiện **check(parseInt,99) == 1835996258**. Nếu thỏa điều kiện, xuất ra dòng

The flag is: và in ra flag với chức năng **stringFromJNI2(parseInt)**. Ngược lại xuất ra **Not Right!**

```
public void onClick(View view) {
    try {
        int parseInt = Integer.parseInt(editText.getText().toString());
        if (MainActivity.this.check(parseInt, 99) == 1835996258) {
            textView.setText("The flag is:");
            textView2.setText("alictf{" + MainActivity.this.stringFromJNI2(parseInt) + "}");
            return;
        }
        textView.setText("Not Right!");
    } catch (NumberFormatException e) {
        textView.setText("Not a Valid Integer number");
    }
}
```

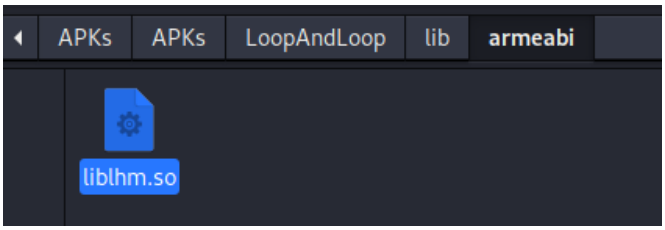
- Tiếp theo ta sẽ xem các hàm check

```
        System.loadLibrary("lhm");
    }

    public native int chec(int i, int i2);

    public int check(int i, int i2) {
        return chec(i, i2);
    }

    public int check1(int i, int i2) {
        for (int i3 = 1; i3 < 100; i3++) {
            i += i3;
        }
        return chec(i, i2);
    }

    public int check2(int i, int i2) {
        int i3 = 1;
        if (i2 % 2 == 0) {
            while (i3 < 1000) {
                i += i3;
                i3++;
            }
            return chec(i, i2);
        }
        while (i3 < 1000) {
            i -= i3;
            i3++;
        }
        return chec(i, i2);
    }

    public int check3(int i, int i2) {
        for (int i3 = 1; i3 < 10000; i3++) {
            i += i3;
        }
        return chec(i, i2);
    }
```

- Ta thấy gồm có 4 hàm check là **check, check1, check2, check3**. Trong các hàm này đều gọi native chec từ thư viện lhm. Kết quả thực hiện của

các vòng lặp có thể dự đoán được, ở hàm check 1 là 1 đến 100, check2 là 1 đến 1000 và check3 là từ 1 đến 10000

- Tiến hành sử dụng IDA để phân tích file **liblhm.so**



- Tại hàm **MainActivity_chec(a1, a2, a3, a4)**

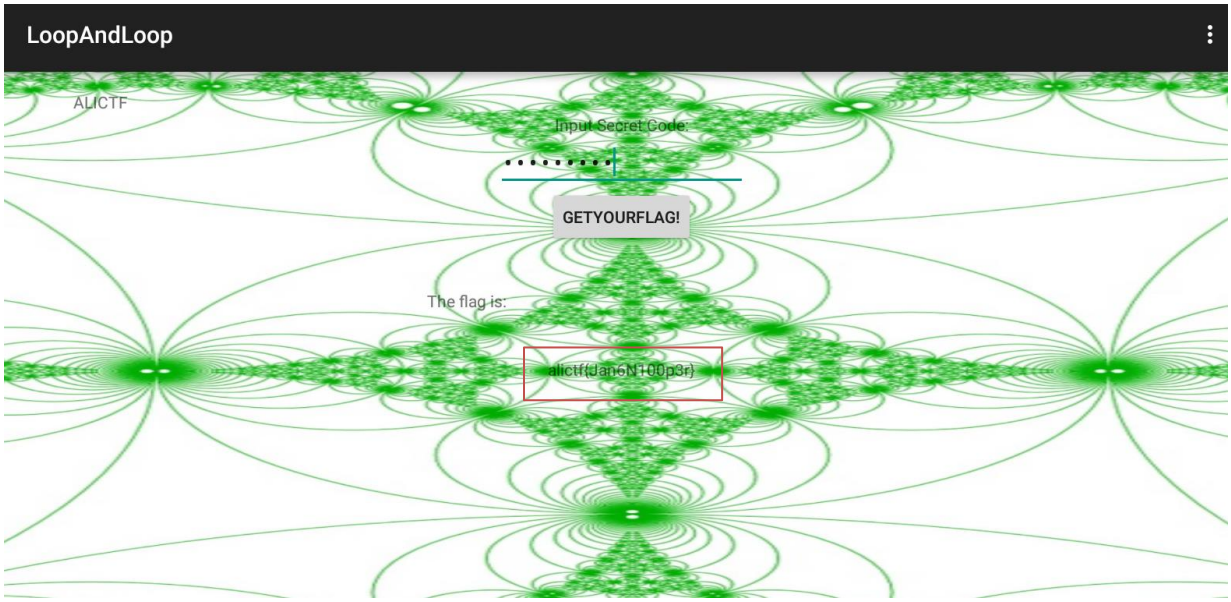
IDA View-A | Pseudocode-A | Strings window | Hex View-1 | Structures | Enums | Imports | Exports

```
1int __fastcall Java_net_bluelotus_tomorrow_easyandroid_MainActivity_chec(int a1, int a2, int a3, int a4)
2{
3  int v5; // r7
4  int result; // r0
5  int v10[9]; // [sp+1Ch] [bp-24h] BYREF
6
7  v5 = (*(int (__fastcall **)(int, const char *))(*(_DWORD *)a1 + 24))(
8         a1,
9         "net/bluelotus/tomorrow/easyandroid/MainActivity");
10 v10[0] = _JNIEnv::GetMethodID(a1, v5, "check1", "(II)I");
11 v10[1] = _JNIEnv::GetMethodID(a1, v5, "check2", "(II)I");
12 v10[2] = _JNIEnv::GetMethodID(a1, v5, "check3", "(II)I");
13 if ( a4 - 1 <= 0 )
14   result = a3;
15 else
16   result = _JNIEnv::CallIntMethod(a1, a2, v10[2 * a4 % 3], a3, a4 - 1);
17 return result;
18}
```

- **JNIEnv::GetMethodID**: Nhận được một ID Phương pháp Java, Chức năng này sẽ trả về các phương pháp của lớp non static hoặc id phương pháp phiên bản giao diện.
- **JNIEnv::CallIntMethod:** Giá trị trả về của hàm là int của phương thức Java.
- Sau khi phân tích, ta có thể biết rằng phương thức kiểm tra gọi ba hàm kiểm tra của lớp Java để xử lý đầu vào của chúng ta theo kết quả nhân tham số thứ hai với 2 và modulo 3 với 3.
- Ta có ý tưởng để tìm ra kết quả ta làm ngược lại thuật toán, làm ngược lại tất cả các hàm.
- Đoạn code thực thi để tìm ra chuỗi số như sau:

```
┌──(kali㉿kali)-[~/…/Bai Tap 10/APKs/APKs/LoopAndLoop]
└─$ cat loopandloop.py
def getinput():
        target = 1835996258
        for i in range(2,100):
                if 2 * i % 3 == 0:
                        target = check1(target,i - 1)
                elif 2 * i % 3 == 1:
                        target = check2(target,i - 1)
                else:
                        target = check3(target,i - 1)
        print(target)

def check1(input,loopNum):
        t = input
        for i in range(1,100):
                t = t - i
        return t

def check3(input,loopNum):
        t = input
        for i in range(1,10000):
                t = t - i
        return t

def check2(input,loopNum):
        t = input
        if loopNum % 2 == 0:
                for i in range(1,1000):
                        t = t - i
                return t
        for i in range(1,1000):
                t = t + i
        return t

if __name__ == '__main__':
        getinput()

┌──(kali㉿kali)-[~/…/Bai Tap 10/APKs/APKs/LoopAndLoop]
└─$ python loopandloop.py
236492408
```

- Chạy và ta thu được kết quả là **236492408**
- Nhập chuỗi này và thực thi ta thu được flag thành công



FLAG: alictf{Jan6N100p3r}