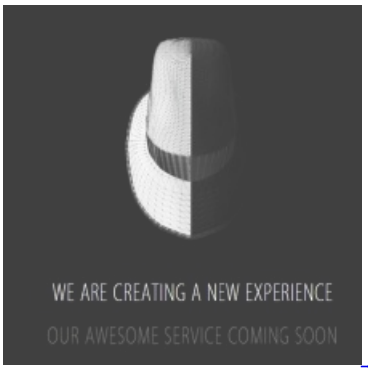
Bugku逆向题目解析

原创

北岸冷若冰霜 于 2020-07-31 20:10:07 发布 713 收藏 3

分类专栏： # CTF夺旗 安全 文章标签： 安全 信息安全 CTF

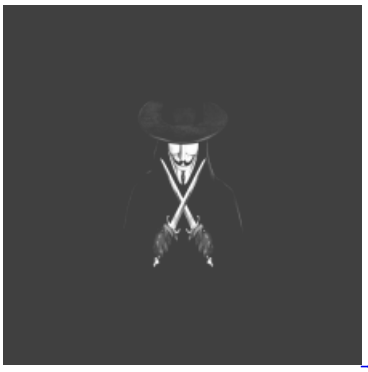
本文链接：https://blog.csdn.net/u013469753/article/details/107720541



CTF夺旗 同时被 2 个专栏收录

7 篇文章 2 订阅

订阅专栏



安全

34 篇文章 3 订阅

订阅专栏

Bugku逆向

目录

Bugku逆向

1.入门逆向

拖入IDA分析

发现有一大串mov，后面的
66h 6ch 61h 67h
为ASCII字符：flag，的16进制表示,下进行
进制转换
https://www.rapidtables.com/convert/number/hex-to-ascii.html
66h6ch61h67h7BH52H65H5FH31H73H5FH53H30H5FH43H30H4FH4CH7DH
得到flag
flag{Re_1s_S0_C0OL}

2.Easy_vb

easy_vb.exe用IDA打开

直接可以看到flag
即：
flag{*N3t_Rev_1s_E4ay*}

3Easy_Re

直接上IDA分析，再hex view中一直翻可以找到

即
DUTCTF{We1c0met0DUTCTF}

4游戏过关

1-8输入一遍？

只要是把1-8每个数字都是有且仅有输过一遍，那就可以得flag。
得到flag即：zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}

5Timer(阿里CTF)

bugKuCTF第四道reverse题目Timer(阿里CTF)writeup

来自 https://blog.csdn.net/xiangshangbashaonian/article/details/80301054
是一道Android逆向分析的题目
jeb反编译，查看MainActivity代码

flag{Y0vAr3TimerMa3te7}

6逆向入门

使用Winhex打开，发现是一张图片，及拿过其复制方法如amdin.txt之后

利用Data URL将小图片生成数据流形式来自 https://blog.csdn.net/pdsu161530247/article/details/75234835
将admin.txt的ASCII码放在admin-image.html中

或者
http://www.vgot.net/test/image2base64.php
bugku{inde_9882ihsd8-0}

扫描得到flag

bugku{inde_9882ihsd8-0}

7.love

是个exe，拖进IDA分析main函数

来自 https://blog.csdn.net/qq_42192672/article/details/82698595

bugkuCTF平台逆向题第五道love题解

来自 https://blog.csdn.net/xiangshangbashaonian/article/details/78923289

flag{i_l0ve_you}

8.LoopAndLoop（阿里CTF）

安卓逆向题目思路Jeb反编译成Java，查看主函数

```
 this.findViewById(2131492946).setOnClickListener(new View$OnClickListener(this.findViewById(2131492944), this.f
indViewById(2131492945), this.findViewById(2131492947)) {
            public void onClick(View arg7) {
                int v1;
                String v2 = this.val$ed.getText().toString();
                try {
                    v1 = Integer.parseInt(v2);
                }
                catch(NumberFormatException v0) {
                    this.val$tv1.setText("Not a Valid Integer number");
                    return;
                }

                if(MainActivity.this.check(v1, 99) == 1835996258) {
                    this.val$tv1.setText("The flag is:");
                    this.val$tv2.setText("alictf{" + MainActivity.this.stringFromJNI2(v1) + "}");
                }
                else {
                    this.val$tv1.setText("Not Right!");
                }
            }
        });
```

关键在check函数的判断如果等于后面数字，则会向flag段落继续执行，然后用stringFromJNI2函数进行变形
如果v1转化数字出现了
异常，直接输出 Not a Valid Integer number
如果对了 就直接输出flag
错了 就输出 not right
现在看看 check 这个函数

两个函数都在native层，把文件里的liblhm.so文件拖出来放到IDA中分析，so文件位于和classes.dex同级别的lib文件夹里

进入IDA文件分析，F5搜索定位字符串,找到主函数

原来的汇编代码

```
result = _JNIEnv::CallIntMethod(v4, v9, *(&v10 + 2 * v8 % 3), v7, v8 - 1);
```

这是个选择语句 选择去check1，2，3

逆向代码1.如下
https://blog.csdn.net/qq_41071646/article/details/83867209

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
int ans=1835996258;
int main()
{
     int j;
    for(int i=99;i>1;i--)
    {
         j=i* 2 % 3;
         if(j==0)
         {
             ans-=4950;
         }
         else if(j==1)
         {
             if((i-1)%2==0)#include <iostream>
```

```
#include <stdio.h>
using namespace std;
int ans=1835996258;
int main()
{
     int j;
    for(int i=99;i>1;i--)
    {
         j=i* 2 % 3;
         if(j==0)
         {
             ans-=4950;
         }
         else if(j==1)
         {
             if((i-1)%2==0)
             ans-=499500;
             else
                ans+=499500;
         }
         else
         {
            ans-=49995000;
         }
    }
    printf("%d\n",ans);
    return 0;
}

             ans-=499500;
             else
                ans+=499500;
         }
         else
         {
            ans-=49995000;
         }
    }
    printf("%d\n",ans);
    return 0;
}
```

或者

逆向代码2如下

https://blog.csdn.net/qq_42192672/article/details/82698595

```python
def check1(input,loop):
    a=input
    for i in range(1,100):
        a=a-i
    return a

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

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

key=1835996258
target = key
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)

#236492408
```

新的思路与代码

https://www.anquanke.com/post/id/84033

LoopAndLoop

用ida调试发现check方法调用chec,会自动调用到check1,第一个参数被传递下去,第二个参数被减一后传递到check1。check1里调用chec函数同理地调用check2,check2里的chec调用check3,check3里的chec调用check1。如此递归下去,发现当第二个参数为1的时候,chec会直接返回参数一的值,也就是递归的边界。估计native的chec函数只起这样的作用,因此不需要逆向liblhm.so了。check1和check3都是加上一个数,check2根据第二个参数的奇偶决定是加上还是删除一个数。因此从最后的检查条件1835996258往前推99步就能推出正确的输入。

```
from ctypes import *
SA = sum(range(100))
SB = sum(range(1000))
SC = sum(range(10000))
S0 = 1835996258
for i in range(1, 99):
    if i % 3 == 0:
        S0 = c_int(S0-SC).value
    if i % 3 == 2:
        S0 = c_int(S0-SA).value
    if i % 3 == 1:
        if i % 2 == 0:
            S0 = c_int(S0-SB).value
        else:
            S0 = c_int(S0+SB).value
print S0
print hex(S0)
for i in range(98, 0, -1):
    if i % 3 == 0:
        S0 = c_int(S0+SC).value
    if i % 3 == 2:
        S0 = c_int(S0+SA).value
    if i % 3 == 1:
        if i % 2 == 0:
            S0 = c_int(S0+SB).value
        else:
            S0 = c_int(S0-SB).value
    print hex(S0)
print S0
```

得到应该输入的数字
236492408
所以flag是
alictf{Jan6N100p3r}

9.逆向-easy100（LCTF）

可参考：
http://www.morecoder.com/article/1088631.html

https://www.jianshu.com/p/28e50996b8c5
先JEB反编译,decompile后，导出工程
（1）查看MainActivity代码

```
protected void onCreate(Bundle arg3) {
    super.onCreate(arg3);
    this.setContentView(2130968602);
    ApplicationInfo v0 = this.getApplicationInfo();
    v0.flags &= 2;
    this.p();
    this.findViewById(2131427413).setOnClickListener(new d(this));
}
```

（2）分析,主函数Main,先执行p函数,p函数后面会用,先不讲.创建了一个按钮监听事件在
classs d中.
class d 中onclick函数,当我们点击安卓的按钮,触发函数.

```
public void onClick(View arg5) {
if(MainActivity.a(this.a, MainActivity.a(this.a), this.a.findViewById(2131427414).getText().toString())) {
View v0 = this.a.findViewById(2131427412);
Toast.makeText(this.a.getApplicationContext(), “Congratulations!”, 1).show();
((TextView)v0).setText(2131099682);
}
else {
Toast.makeText(this.a.getApplicationContext(), “Oh no.”, 1).show();
}
}
```

if判断正确就显示flag.调用了Main函数中的a函数,第一个参数是句柄,第二个参数是调用了a函数(另外一个)返回一个字符串,第三个参数是我们输入的字符串.

（3）a函数分析

```
private String v;

static String a(MainActivity arg1) {
    return arg1.v;
}
```

通过调用a,返回Main中的字符串v,字符串v的初始化在p函数中进行.

（4）p函数分析

```
private void p() {
        try {
            InputStream v0_1 = this.getResources().getAssets().open("url.png");
            int v1 = v0_1.available();
            byte[] v2 = new byte[v1];
            v0_1.read(v2, 0, v1);
            byte[] v0_2 = new byte[16];
            System.arraycopy(v2, 144, v0_2, 0, 16);
            this.v = new String(v0_2, "utf-8");
        }
        catch(Exception v0) {
            v0.printStackTrace();
        }
    }
```

p函数的作用就是读取一张图片的二进制数据取出这张图片byte[144:144+16]的数据保存在v字符串中.

（5）继续分析if判断语句

上面说了if语句调用了Main的a函数(三个参数)

```
static boolean a(MainActivity arg1, String arg2, String arg3) {
        return arg1.a(arg2, arg3);
    }

    private boolean a(String arg4, String arg5) {
        return new c().a(arg4, arg5).equals(new String(new byte[]{21, -93, -68, -94, 86, 117, -19, -68, -92, 33,
 50, 118, 16, 13, 1, -15, -13, 3, 4, 103, -18, 81, 30, 68, 54, -93, 44, -23, 93, 98, 5, 59}));
    }
```

可以看出a函数(三个参数)的调用了a函数(两个参数).而a函数(两个参数)的调用c的a函数(两个参数).计算完后和后面的字节比较,如果相等就显示flag.

（6）a函数（两个参数）分析

```
public String a(String arg5, String arg6) {
    String v0 = this.a(arg5);
    String v1 = "";
    a v2 = new a();
    v2.a(v0.getBytes());
    try {
        v0 = new String(v2.b(arg6.getBytes()), "utf-8");
    }
    catch(Exception v0_1) {
        v0_1.printStackTrace();
        v0 = v1;
    }

    return v0;
}
```

和

```
this.a = new SecretKeySpec(arg4, "AES");
            this.b = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

arg5是从图片中获取的字符串,arg6是我们输入的.
后面代码主要就是AES加密.将arg5经过变换后的字符串当做密码,将输入的字符串进行AES加密后和后面给出的字符串比较,如果相等得到flag.

总结算法
题目主要考察AES加密，ECB模式，PKCS5Padding填充，密钥为MainActivity中成员v的变化，明文为在app中输入的内容。然后AES加密结果和byte数组转化的String比较。若相同则弹出“Congratulation”，说明输入的即为flag。

来自 http://www.morecoder.com/article/1088631.html

那我们就可以直接解密AES就可以得到flag了.
脚本见getkey.py
得到密码的16进制,然后解密AES解密得到flag
LCTF{1t’s_rea1ly_an_ea3y_ap4}

10 SafeBox（NJCTF）

代码直接放Jeb进行反汇编导出代码
参考
https://blog.csdn.net/Ni9htMar3/article/details/66970006
https://www.cnblogs.com/zhengjim/p/10002657.html#safeboxnjctf

看到onCreate方法关键位置18行-37行，输入一个8位数满足条件后，将其变换后与NJCTF{和f4n}拼接
这个8位是一个回文数，并且限制比较具体。
使用python脚本爆破
脚本内容见flag-get.py

计算出回文数i=48533584
flag为：NJCTF{05#f4n}
但提交却提示错误，继续查看androidTest类

这个不限制中间两位必须相等，而且后面有+10，继续使用脚本爆破，脚本内容见flag-get2.py
得到两个结果
48533584
NJCTF{have05-f4n}
48539584
NJCTF{have05if4n}
第二个正确
NJCTF{have05if4n}

11Mountain climbing

运行缺少组件，就没去单独找组件。
下载工具
https://bbs.pediy.com/thread-152454.htm
用PEID查看，发现有UPX壳

52pojie下载脱壳工具
https://www.52pojie.cn/thread-647313-1-1.html

12Take the maze

https://blog.csdn.net/qq_19861715/article/details/79403986

用PEID查看，无壳

放到IDA分析