

JACTF baby_reverse writeup

原创

qq_43710556 于 2019-07-25 22:15:00 发布 276 收藏

分类专栏: [CTF](#) 文章标签: [CTF逆向学习](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_43710556/article/details/97291260

版权



[CTF 专栏收录该内容](#)

5 篇文章 0 订阅

订阅专栏

JACTF baby_reverse writeup

最近一直在学习逆向,所以把自己做过的题记录下来.

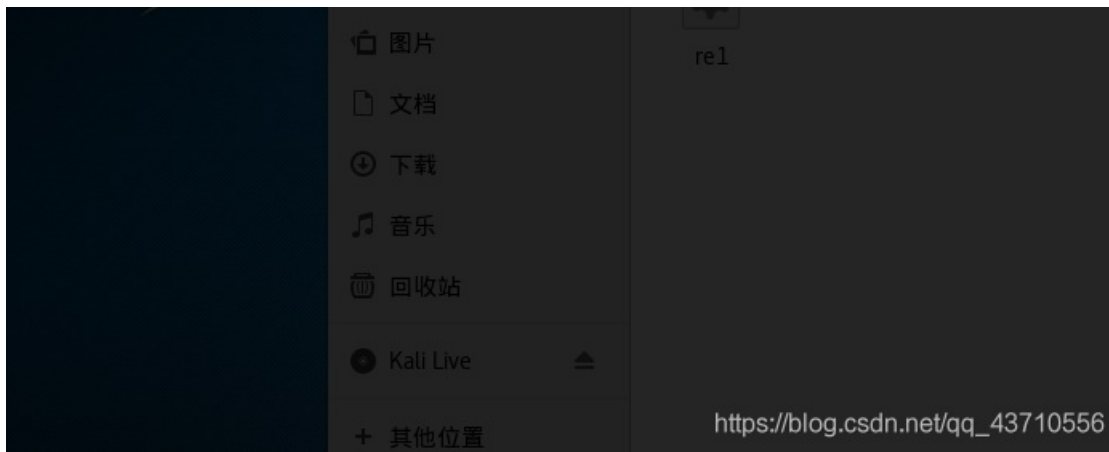
发现jactf的题还不错, 所以就从jactf开始刷题吧。。

```
root@kali: ~/ida
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~/ida# file '/root/ida/baby_reverse'
/root/ida/baby_reverse: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV
), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux
3.2.0, BuildID[sha1]=1bcf1f307c31382f36ffa5532ea4348f6812b0db, not stripped
root@kali:~/ida#
```

下载之后查看一下文件类型, 发现是64位elf文件

放到我的64位kali里面运行一下看看

```
root@kali: ~/ida
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~/ida# './root/ida/baby_reverse'
Please Input Key: 123
Your Length is Wrong
root@kali:~/ida#
```



发现是输入Key，然后程序判断对错。

接下来放IDA里面看看程序的内容。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char s; // [rsp+0h] [rbp-1E0h]
    char v5; // [rsp+F0h] [rbp-F0h]

    memset(&s, 0, 0x1EuLL);
    printf("Please Input Key: ", 0LL);
    __isoc99_scanf("%s", &v5);
    encode(&v5, &s);
    if ( !strcmp(&s, enflag) )
        puts("You are Right");
    return 0;
}
```

https://blog.csdn.net/qq_43710556

发现主函数很简单，

大体的意思就是先初始化一个数组，这个数组的长度是0x1E，并且全部赋值为0。

之后就是输入Key，然后有一个encode函数，看名字是一个加密函数，

最后再将加密后的Key和enflag比较。

大体意思明白了，我们就先看看这个enflag是什么。

双击点击enflag。

```
.data:00000000000004058 __dso_handle dq offset __dso_handle ; DATA XREF: __do_global_dtors_aux+171
.data:00000000000004058 ; .data:__dso_handle↓o
.data:00000000000004060 public enflag
.data:00000000000004060 ; char enflag[]
.data:00000000000004060 enflag db 'bIwhroo8cwqgwruxsi',0
.data:00000000000004060 ; DATA XREF: main+70↑o
.data:00000000000004073 db 0
.data:00000000000004074 db 0
.data:00000000000004075 db 0
.data:00000000000004076 db 0
.data:00000000000004077 db 0
.data:00000000000004078 db 0
.data:00000000000004079 db 0
.data:0000000000000407A db 0
.data:0000000000000407B db 0
.data:0000000000000407C db 0
.data:0000000000000407D db 0
.data:0000000000000407D _data ends
.data:0000000000000407D
```

https://blog.csdn.net/qq_43710556

看到enflag='blwhroo8cwqgwrxusi'

之后我们再看看encode函数.

```

int __fastcall encode(const char *a1, __int64 a2)
{
    char v3[32]; // [rsp+10h] [rbp-70h]
    char v4[32]; // [rsp+30h] [rbp-50h]
    char v5[36]; // [rsp+50h] [rbp-30h]
    int v6; // [rsp+74h] [rbp-Ch]
    int v7; // [rsp+78h] [rbp-8h]
    int i; // [rsp+7Ch] [rbp-4h]

    v7 = 18;
    i = 0;
    v6 = 0;
    if ( strlen(a1) != 18 )
        return puts("Your Length is Wrong");
    puts("flag{This_1s_f4cker_flag}");
    for ( i = 0; i < v7; i += 3 )
    {
        v5[i] = v7 ^ (a1[i] + 6);
        v4[i + 1] = (a1[i + 1] - 6) ^ v7;
        v3[i + 2] = a1[i + 2] ^ 6 ^ v7;
        *(_BYTE *)(a2 + i) = v5[i];
        *(_BYTE *)(a2 + i + 1LL) = v4[i + 1];
        *(_BYTE *)(a2 + i + 2LL) = v3[i + 2];
    }
    return a2;
}

```

https://blog.csdn.net/qq_43710556

这里其实有个小坑。。。当你在程序中输入上面的enflag后会出现



https://blog.csdn.net/qq_43710556

让你以为是flag...其实是个假flag(打ctf英语真是得学好, faker...假的)

仔细看看这个encode函数, 加密的过程就在for循环中,

我们要做的就是如何写出decode解密函数, 把加密过程逆回来。

仔细看这个循环。

```

for ( i = 0; i < v7; i += 3 )
{
    v5[i] = v7 ^ (a1[i] + 6);
    v4[i + 1] = (a1[i + 1] - 6) ^ v7;
    v3[i + 2] = a1[i + 2] ^ 6 ^ v7;
}

```

```

v4[i + 1] = (a1[i + 1] - 0) ^ v7,
v3[i + 2] = a1[i + 2] ^ 6 ^ v7;
*(_BYTE *) (a2 + i) = v5[i];
*(_BYTE *) (a2 + i + 1LL) = v4[i + 1];
*(_BYTE *) (a2 + i + 2LL) = v3[i + 2];
}
return a2;
}

```

https://blog.csdn.net/qq_43710556

从后边开始分析。

返回的a2就是经过加密的enflag

所以a2='blwhroo8cwqgwrxusi'

for (i=0;i<v7;i+=3)

每循环一次i就加3

而每次循环a2的值其实都是由v5,v4,v3控制的

也就是可以吧a2分组，3个字符一组。

v5就是a2[i]

v4就是a2[i+1]

v3就是a2[i+2]

分析到这里，就已经可以吧v5,v4,v3这三个字符串是什么求出来了。

上python:

```

enflag='blwhroo8cwqgwrxusi'
deflag=''
v3=''
v4=''
v5=''
for i in range(0,len(enflag),3):
    v5+=enflag[i]
    v4+=enflag[i+1]
    v3+=enflag[i+2]

```

https://blog.csdn.net/qq_43710556

因为异或是可逆的。

加密前^密钥=加密后

加密后^密钥=加密前

所以我们可以逆出a2加密前的原始字符串

其实就是把v5,v4,v3这三个字符串逆回去

最终的decode脚本就是:

```

enflag='blwhroo8cwqgwrxusi'
deflag=''
v3=''
v4=''
v5=''
for i in range(0,len(enflag),3):
    v5+=enflag[i]
    v4+=enflag[i+1]
    v3+=enflag[i+2]
for i in range(0,6):
    deflag+=chr((ord(v5[i])^18)-6)
    deflag+=chr((ord(v4[i+1])^18)+6)

```

```
deflag+=chr((ord(v4[1])^18)+6)
deflag+=chr(ord(v3[i])^(6^18))
print(deflag)
```

https://blog.csdn.net/qq_43710556

flag就不放出了。。

这题其实也不难，考验的就是静态分析。

这题我也是做了好久，才分析清楚。

刚开始学习逆向，加油，明天继续。