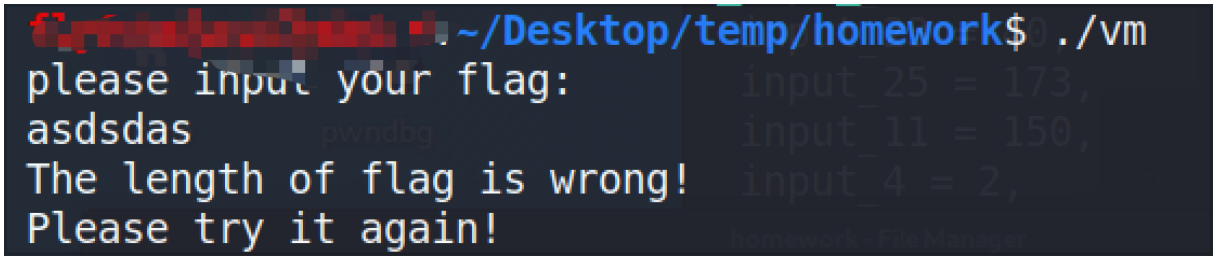
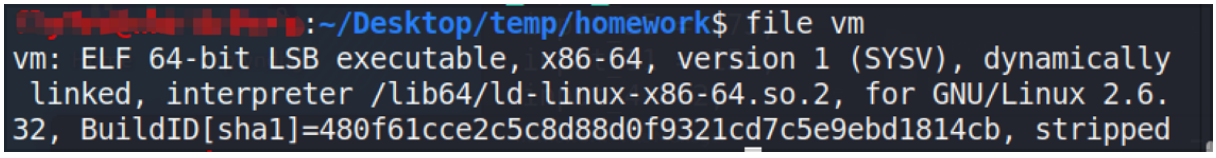
2020年10月7日 23:51 VM [已上传]

程序名 哇VM



一道 Linux 逆向

输入 flga 错误



64位的Linux程序

使用64 位IDA载入

找到main函数代码如下

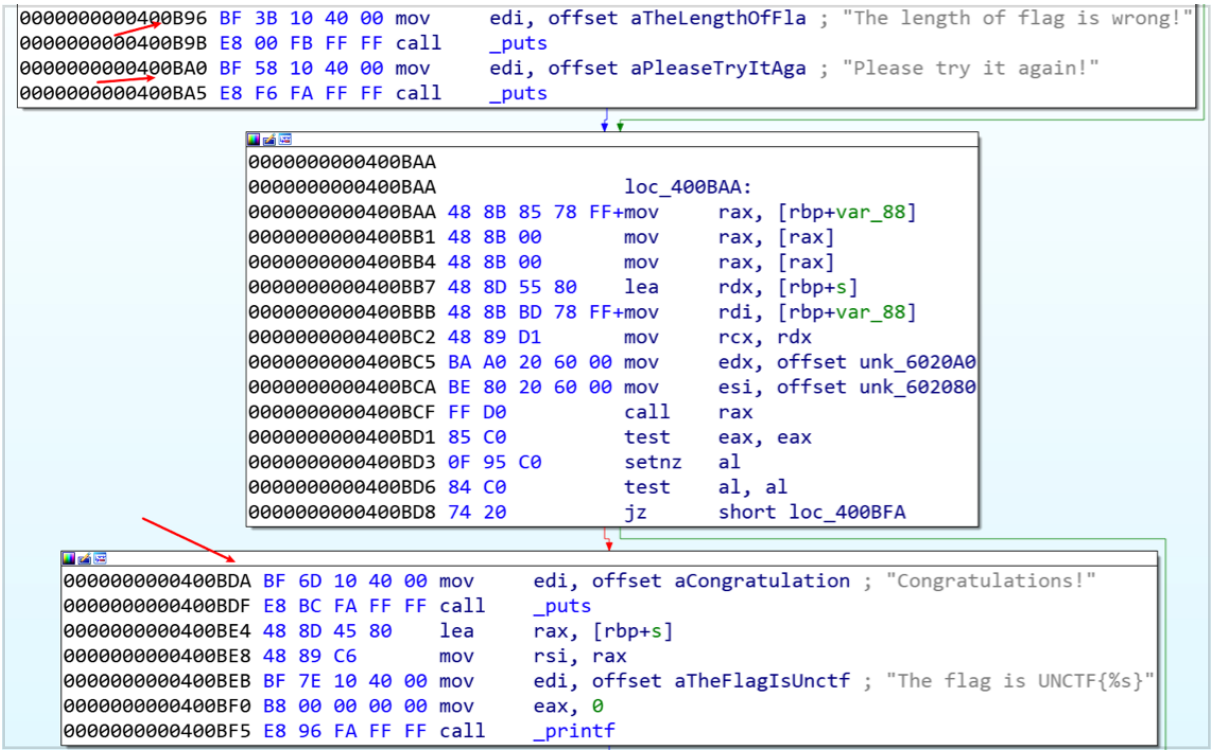
```
signed __int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  unsigned int (__fastcall ***v3)(_QWORD, void *, void *, char *); // rbx
  char s; // [rsp+10h] [rbp-80h]
  int v6; // [rsp+70h] [rbp-20h]
  unsigned __int64 v7; // [rsp+78h] [rbp-18h]

  v7 = __readfsqword(0x28u);
  memset(&s, 0, 0x60uLL);
  v6 = 0;
  v3 = (unsigned int (__fastcall ***)(_QWORD, void *, void *, char *))operator
new(0x28uLL);
  sub_400C1E(v3, a2);
  puts("please input your flag:");
  scanf("%s", &s);
  if ( strlen(&s) != 32 )
  {
    puts("The length of flag is wrong!");
    puts("Please try it again!");
  }
```

```
  if ( (**v3)(v3, &unk_602080, &unk_6020A0, &s) != 0 )
  {
    puts("Congratulations!");
    printf("The flag is UNCTF{%s}", &s);
  }
  return 1LL;
}
```

flag长度为32位 具体判断在 operator new(ulong)

虚拟机指令 较复杂，尝试使用Angr 进行求解



写出对应的angr 脚本

```
import angr
proj = angr.Project("easyvm",auto_load_libs=False)       #easyvm为程序名
simgr = proj.factory.simgr()
avoid_list=[0x400BA0,0x400B96]        #avoid_list为需要避开的地址，如“失败”“错误”等等，需要用IDA等手动查看
simgr.explore(find=0x400BDA,avoid=avoid_list)       #find为需要到达的地址，如“成功”“正确”等等，需要用IDA等手动查看
print(simgr.found[0].posix.dumps(0))
```

```
import angr
proj = angr.Project("vm",auto_load_libs=False)
simgr = proj.factory.simgr()
avoid_list=[0x400BA0,0x400B96]
simgr.explore(find=0x400BDA,avoid=avoid_list)
print(simgr.found[0].posix.dumps(0))
```

```
ents/Angr$ python3 exp.py
b'942a4115be2359ffd675fa6338ba23b6\x00\xc5\x19\x01\x01\x01)\x00\x1d\x00\x00\x02J\x00I\x00\x00$)\x00\x01'
x00\x02)\x00\x00'
```

爆破的速度较慢，最后得到flag

```
please input your flag:
942a4115be2359ffd675fa6338ba23b6
Congratulations!
The flag is UNCTF{942a4115be2359ffd675fa6338ba23b6}
```

UNCTF{942a4115be2359ffd675fa6338ba23b6}

此题的另一种解法

还原出main函数对应的源码：

```
#include<stdio.h>
int main(){
int i;
unsigned char a16=0,a17=0;
char input[]="66666666666666666666666666666666";
unsigned char c[32]={0xF4,0x0A,0xF7,0x64,0x99,0x78,0x9E,0x7D,
            0xEA,0X7B,0X9E,0X7B,0X9F,0X7E,0XEB,0X71,
            0XE8,0X00,0XE8,0X07,0X98,0X19,0XF4,0X25,
            0XF3,0X21,0XA4,0X2F,0XF4,0X2F,0XA6,0X7C};
for(i=0;i<32;i++){
  a16=input[i];
  a16-=i;
  a17=a16 ^ a17;
  a16=-51;
  a16=a16^a17;
  if(a16==c[i]){
    puts("YES");
    a17=a16;
  }
```

```
  else{
    puts("NO");
    break;
  }
}
return 0;
}
```

写出相应的解密脚本：

```
#include<stdio.h>
int main(){
int i;
unsigned char a16=0,a17=0;
unsigned char c[32]={0xF4,0x0A,0xF7,0x64,0x99,0x78,0x9E,0x7D,
            0xEA,0X7B,0X9E,0X7B,0X9F,0X7E,0XEB,0X71,
            0XE8,0X00,0XE8,0X07,0X98,0X19,0XF4,0X25,
            0XF3,0X21,0XA4,0X2F,0XF4,0X2F,0XA6,0X7C};
printf("9");
for(i=1;i<32;i++){
printf("%c",(c[i]^(-51)^c[i-1])+i);
}
return 0;
}
```

UNCTF{942a4115be2359ffd675fa6338ba23b6}

Reference

博客 https://www.cnblogs.com/0xJDchen/p/9291335.html

angr官方代码和文档在github

https://github.com/angr/angr

动若脱兔:深入浅出angr—初步理解符号执行以及angr架构 - 0xJDchen - 博客园

见微知著(一)：解析ctf中的pwn--Fast bin里的UAF - 0xJDchen - 博客园

遇到的问题

pip install angr

报错

```
ERROR: Could not install packages due to an EnvironmentError:
HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Max retries exceeded
with url: /packages/4e/5f/
528232275f6509b1fff703c9280e58951a81abe24640905de621c9f81839/pip-20.2.3-py2.py3-
none-any.whl (Caused by
NewConnectionError('<pip._vendor.urllib3.connection.VerifiedHTTPSConnection object
at 0x7fb0c6b29340>: Failed to establish a new connection: [Errno -2]
```

python -m install –upgrade pip

#CTF #符号执行