# 攻防世界-logmein-wp

原创

qlykldog 于 2021-10-16 18:47:40 发布 56 收藏

文章标签： c语言



本文链接：https://blog.csdn.net/m0_54782683/article/details/120802328

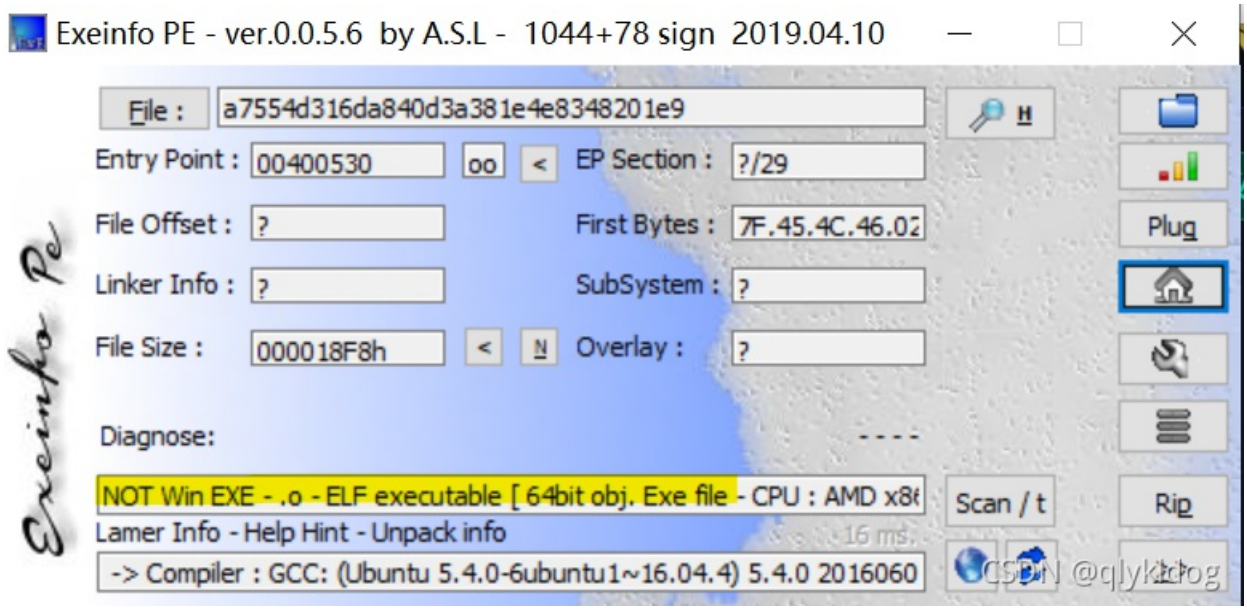


## 攻防世界-logmein-wp

今天给大家带来的是一道简单算法逆向题目，正如题目提示所说“菜鸡开始接触一些基本的算法逆向了”。那我们直接开始。

逆向第一步：把下载得到的文件拖进Exeinfo PE里，看到高亮部分：

`NOT Win EXE - .o - ELF executable [ 64bit obj. Exe file` 。。。




说明这是一个ELF文件，那么通过之前的题目我们就可以知道将其放入Linux环境中就可以运行，是个64位elf，那么放进IDA Pro。（我把文件名改成了hh，图个简单）

```
qlykldog@qlykldog-virtual-machine:~/桌面$ file hh
hh: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32
, BuildID[sha1]=c8f7fb137d9be24a19eb4f10efc29f7a421578a7, stripped
```

当然我们也可以直接把他放进64位IDA Pro中。

依旧是老步骤，shift+F12查看全部字符串，找到关键字符

Enter your guess:

| Address | Length | Type | String |
|---|---|---|---|
| LOAD:000000... | 0000001C | C | /lib64/ld-linux-x86-64.so.2 |
| LOAD:000000... | 0000000A | C | libc.so.6 |
| LOAD:000000... | 0000000F | C | __isoc99_scanf |
| LOAD:000000... | 00000007 | C | printf |
| LOAD:000000... | 00000007 | C | strlen |
| LOAD:000000... | 00000012 | C | __libc_start_main |
| LOAD:000000... | 0000000F | C | __gmon_start__ |
| LOAD:000000... | 0000000A | C | GLIBC_2.7 |
| LOAD:000000... | 0000000C | C | GLIBC_2.2.5 |
| .rodata:00000... | 0000002D | C | Welcome to the RC3 secure password guesser.\n |
| .rodata:00000... | 00000033 | C | To continue, you must enter the correct password.\n |
| .rodata:00000... | 00000013 | C | Enter your guess: |
| .rodata:00000... | 00000015 | C | Incorrect password!\n |
| .rodata:00000... | 0000002E | C | You entered the correct password!\nGreat job!\n |
| .eh_frame:000... | 00000006 | C | ;*3$\" |



点击，出现下图。

```
.rodata:0000000000400938 aEnterYourGuess db 'Enter your guess: ',0
.rodata:0000000000400938                                         ; DATA XREF: main+6E↑o
.rodata:000000000040094B a32s            db '%32s',0             ; DATA XREF: main+82↑o
.rodata:0000000000400950 ; const char aIncorrectPassw[]
```

选中main后，按Ctrl+X交叉引用。

xrefs to aEnterYourGuess

| Directio | Ty | Address | Text |
|---|---|---|---|
| Up | o | main+6E | mov rdi, offset aEnterYourGuess; "Enter your guess: " |

Line 1 of 1

OK Cancel Search Help



点击OK跳转，我们就来到了main函数部分。

```
mov     [rbp+var_10], cx
mov     rax, ds:qword_4008D0
mov     [rbp+var_28], rax
mov     [rbp+var_2C], 7
mov     al, 0
call    _printf
mov     rdi, offset aToContinueYouM ; "To continue, you must enter the correct"...
mov     [rbp+var_5C], eax
mov     al, 0
call    _printf
mov     rdi, offset aEnterYourGuess ; "Enter your guess: "
```



按一下F5，可以查看伪C代码。

```
void __fastcall __noreturn main(int a1, char **a2, char **a3)
{
  size_t v3; // rsi
  int i; // [rsp+3Ch] [rbp-54h]
  char s[36]; // [rsp+40h] [rbp-50h] BYREF
  int v6; // [rsp+64h] [rbp-2Ch]
  __int64 v7; // [rsp+68h] [rbp-28h]
  char v8[28]; // [rsp+70h] [rbp-20h] BYREF
  int v9; // [rsp+8Ch] [rbp-4h]

  v9 = 0;
  strcpy(v8, ":\"AL_RT^L*.?+6/46");
  v7 = 0x65626D61726168LL;
  v6 = 7;
  printf("Welcome to the RC3 secure password guesser.\n");
  printf("To continue, you must enter the correct password.\n");
  printf("Enter your guess: ");
  __isoc99_scanf("%32s", s);
  v3 = strlen(s);
  if ( v3 < strlen(v8) )
    sub_4007C0();
  for ( i = 0; i < strlen(s); ++i )
  {
    if ( i >= strlen(v8) )
      sub_4007C0();
    if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
      sub_4007C0();
  }
  sub_4007F0();
}
```

我们来分析一下main函数的整个运算逻辑。

将一个奇怪的字符串赋值给v8：

```
strcpy(v8, ":\"AL_RT^L*.?+6/46");
```

我们输入的东西放在s里：

```
__isoc99_scanf("%32s", s);
```

v3是计算我们输入的字符串的长度：

```
v3 = strlen(s);
```

接下来是一个判断语句：

```
if ( v3 < strlen(v8) )
    sub_4007C0();
```

如果我们输入的字符串长度小于v8那串奇怪的字符串的长度，执行`sub_4007C0()`。点开`sub_4007C0()`，果不其然是判断错误。

```
1 void __noreturn sub_4007C0()
2 {
3   printf("Incorrect password!\n");
4   exit(0);
5 }
```

如果长度大于或等与v8则进入下面的循环，再看看接下来的for循环语句：

```
for ( i = 0; i < strlen(s); ++i )
  {
    if ( i >= strlen(v8) )
      sub_4007C0();
    if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
      sub_4007C0();
  }
```

如果输入的字符串和经过运算后的后字符串不等，则进入sub_4007c0，还是输出Incorrect password。

我们可以看到sub_4007F0()里是我们想要的东西：

```
void __noreturn sub_4007F0()
{
  printf("You entered the correct password!\nGreat job!\n");
  exit(0);
}
```

那么我们只需要使`s[i] == (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i])`就可以进入`sub_4007F0()`函数。

于是接下来我么就可以写脚本啦：

（c++脚本）

```
#include<bits/stdc++.h>
using namespace std;
char v8[8];
long long  v7 = 0x65626D61726168LL;
int main()
{
 char *a=(char *)&v7;//将v7转化为字符串
 strcpy(v8,":\"AL_RT^L*.?+6/46");
 for(int i=0;i<17;i++)
  v8[i]^=a[i%7];
 cout<<v8;
 return 0;
}
```

（python脚本）

```
key = ':\"AL_RT^L*.?+6/46'
v7 = 'ebmarah'
v7 = v7[::-1]
v8 = len(key)
result = ''
for i in range(v8):
 result += chr((ord(v7[i % 7]) ^ ord(key[i])))
print(result)
```

注意：我们之前可以在IDA中看到 v7 = 0x65626D61726168LL;我么选中该16进制数字，再按R键，就可以变成字符串v7 = ‘ebmarah’。最重要的是：由于程序是小段的存储方式，所以，ebmarah就得变成harambe。

那么反序，是为什么呢？

我们可以在hex view窗口看到这里的存储方式是小端存储，所以我们需要反序。

> x86系列的CPU都是以小端序储存数据的，即低位字节存入低地址，高位字节存入高地址。

```
0004008D0  68 61 72 61 6D 62 65 00  57 65 6C 63 6F 6D 65 20  harambe.Welcome·
0004008E0  74 6F 20 74 68 65 20 52  43 33 20 73 65 63 75 72  to·the·RC3·secur
0004008F0  65 20 70 61 73 73 77 6F  72 64 20 67 75 65 73 73  e·password·guess
000400900  65 72 2E 0A 00 54 6F 20  63 6F 6E 74 69 6E 75 65  er...To·continue
000400910  2C 20 79 6F 75 20 6D 75  73 74 20 65 6E 74 65 72  ,·you·must·enter
000400920  20 74 68 65 20 63 6F 72  72 65 63 74 20 70 61 73  ·the·correct·pas
000400930  73 77 6F 72 64 2E 0A 00  45 6E 74 65 72 20 79 6F  sword...Enter·yo
```

> ord():是将字符串转换为ascii格式
> chr():是将ascii转换为字符串

那么就可以得到最后的flag啦：

```
flag{RC3-2016-XORISGUD}
```