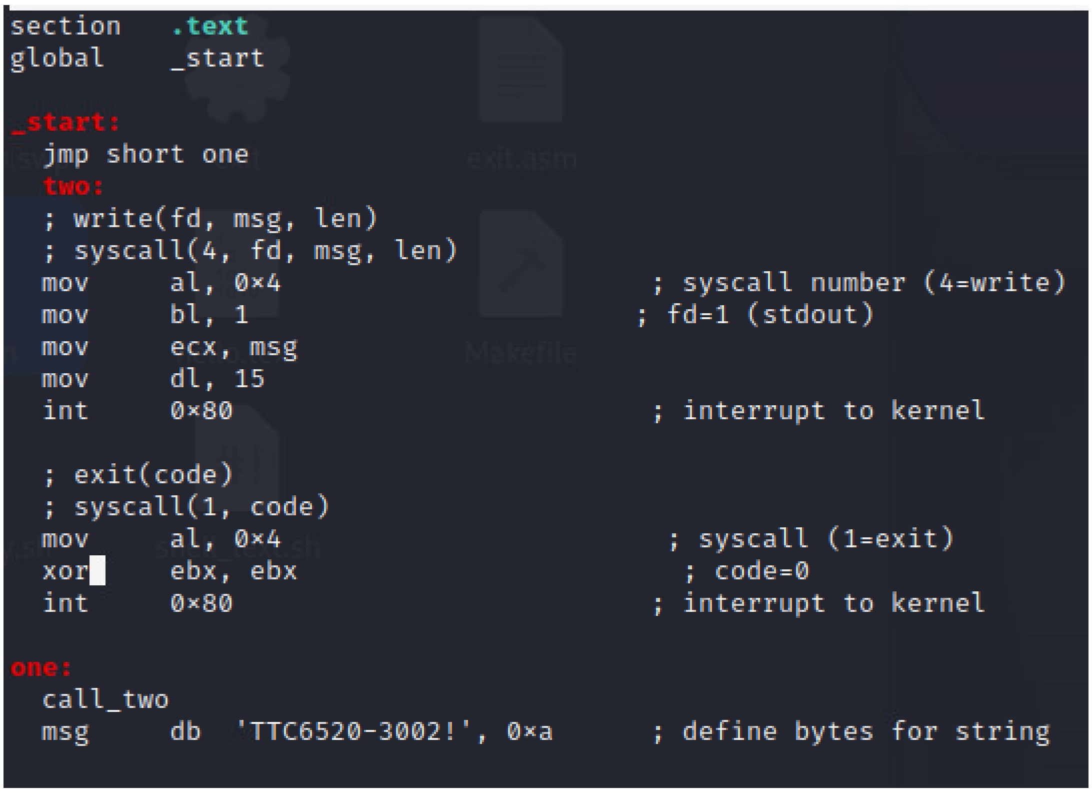
Kalle Jalkanen AB8822

Introduction

The purpose of this assignment is to generate working Shellcode and test it.

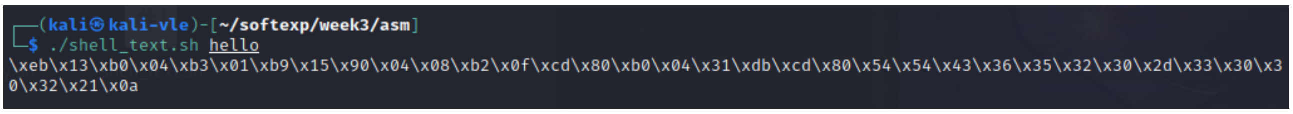
Working steps

I started with removing null bytes with ‘0x4’ from the registers and used low registers instead of extended. Then I also xored ebx register to be zero.

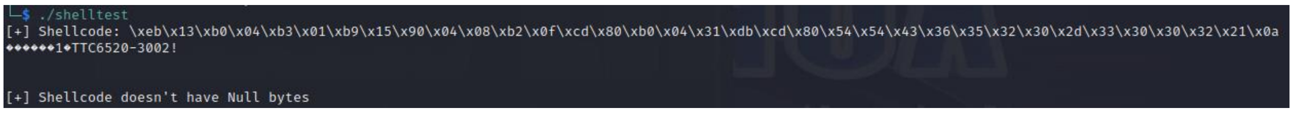


```
section   .text
global    _start

_start:
  jmp short one
  two:
  ; write(fd, msg, len)
  ; syscall(4, fd, msg, len)
  mov      al, 0×4                        ; syscall number (4=write)
  mov      bl, 1                         ; fd=1 (stdout)
  mov      ecx, msg
  mov      dl, 15
  int      0×80                           ; interrupt to kernel

  ; exit(code)
  ; syscall(1, code)
  mov      al, 0×4                         ; syscall (1=exit)
  xor      ebx, ebx                          ; code=0
  int      0×80                           ; interrupt to kernel

one:
  call_two
  msg      db   'TTC6520-3002!', 0×a      ; define bytes for string
```

Then I ran make to the hello.asm and ran it through shell_text.sh. Then I tested it with shelltest and I said that shellcode doesn’t have null bytes.

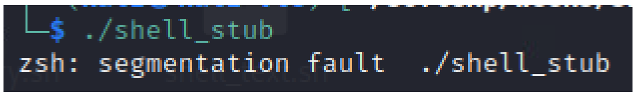


```
┌──(kali㉿kali-vle)-[~/softexp/week3/asm]
└─$ ./shell_text.sh hello
\xeb\x13\xb0\x04\xb3\x01\xb9\x15\x90\x04\x08\xb2\x0f\xcd\x80\xb0\x04\x31\xdb\xcd\x80\x54\x54\x43\x36\x35\x32\x30\x2d\x33\x30\x3
0\x32\x21\x0a
```



```
└─$ ./shelltest
[+] Shellcode: \xeb\x13\xb0\x04\xb3\x01\xb9\x15\x90\x04\x08\xb2\x0f\xcd\x80\xb0\x04\x31\xdb\xcd\x80\x54\x54\x43\x36\x35\x32\x30\x2d\x33\x30\x30\x32\x21\x0a
◆◆◆◆◆◆1◆TTC6520-3002!


[+] Shellcode doesn't have Null bytes
```

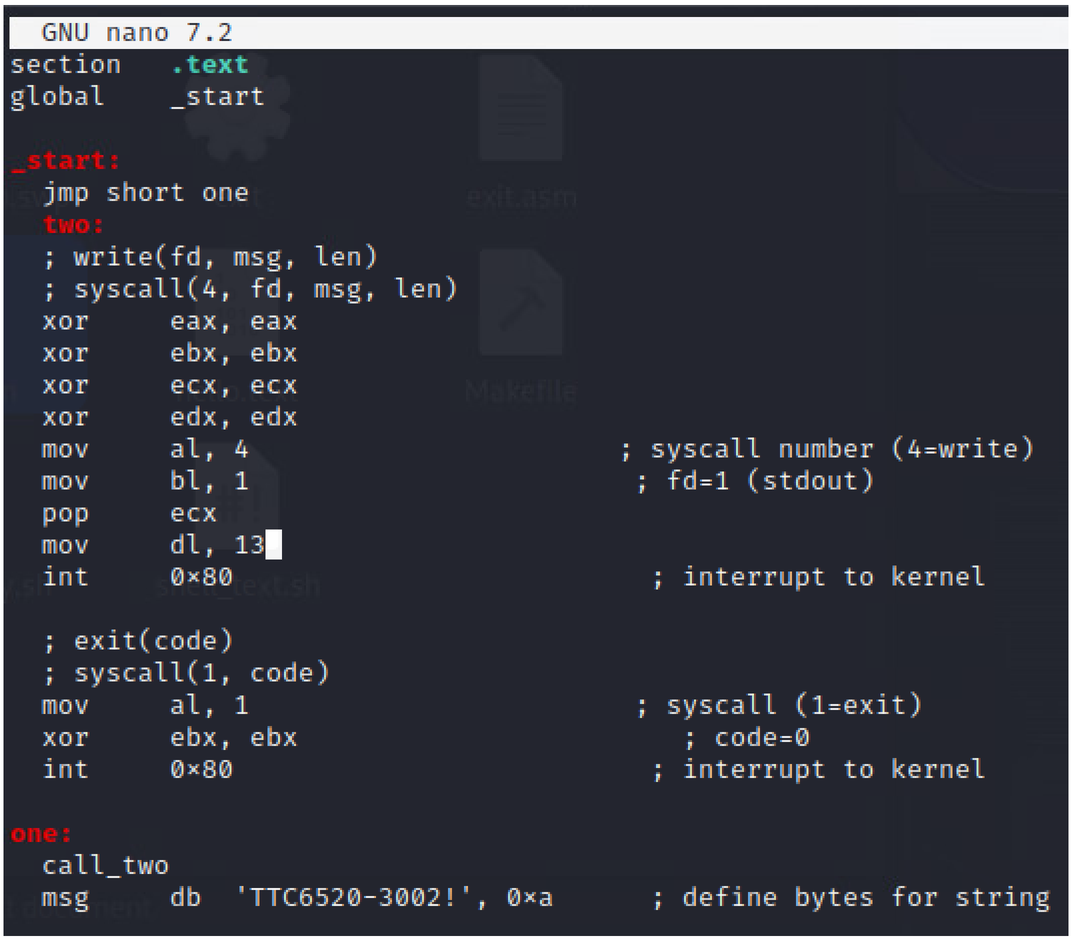
This was all I could successfully do and after this I just tested all kind of things but did not make any progress. Next, I tried to xor all registers to be 0, instead of moving msg to ecx pop ecx and tried to change ‘mov edx, msglen’ to move messages length to dl. Then I ran hello through shell_array.sh and copied it to shell_stub.c and compiled it. I ran shell_stub but it gave segmentation fault.



```
└─$ ./shell_stub
zsh: segmentation fault  ./shell_stub
```

Results

At the end I only get trough the first step of the assignment and I used about 4 hours to this. I have changed hello.asm so much that I can't remember which attempts are genuine and which are just random tests. In the end, it looks like this.

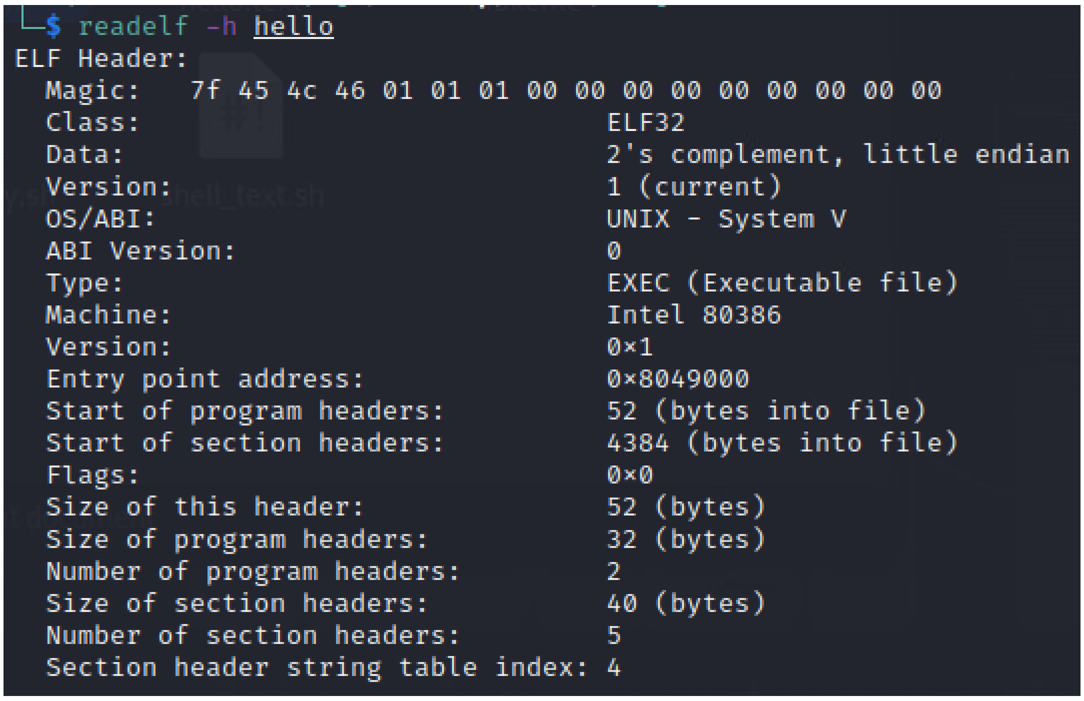


```
 GNU nano 7.2
section   .text
global    _start

_start:
  jmp short one
  two:
  ; write(fd, msg, len)
  ; syscall(4, fd, msg, len)
  xor      eax, eax
  xor      ebx, ebx
  xor      ecx, ecx
  xor      edx, edx
  mov      al, 4                          ; syscall number (4=write)
  mov      bl, 1                           ; fd=1 (stdout)
  pop      ecx
  mov      dl, 13
  int      0x80                             ; interrupt to kernel

  ; exit(code)
  ; syscall(1, code)
  mov      al, 1                           ; syscall (1=exit)
  xor      ebx, ebx                            ; code=0
  int      0x80                             ; interrupt to kernel

one:
  call two
  msg      db   'TTC6520-3002!', 0xa        ; define bytes for string
```

System information

Linux kali-vle 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64 GNU/Linux

Program was compiled with Makefile.



```
└─$ readelf -h hello
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0×1
  Entry point address:               0×8049000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          4384 (bytes into file)
  Flags:                             0×0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         2
  Size of section headers:           40 (bytes)
  Number of section headers:         5
  Section header string table index: 4
```