

Lab 2

Main Goals

- Develop Assembly programs:
 - Print a string on the screen
 - Spawn a new shell using `execve`
- Get familiar with two techniques: relative addressing and pushing data into the stack.
- To get familiar with one technique to build a *working* shellcode (more details next lab).

Activity 1: Print on Screen

- Startup code is provided for `print_rel.asm` and `print_stk.asm`
- You need to:
 - Complete the missing parts
 - Answer few questions about the program

Activity 1: Relative Addressing

```
_start:
    ??                ; (complete)

shellcode:
    ??                ; (complete)
    mov eax, ??       ; (complete) opcode for write system call
    mov ebx, ??       ; (complete) 1st arg is the fd
    mov ecx, ??       ; (complete) 2nd arg is the str address
    mov edx, 15       ; 3rd arg is len
    int 0x80          ; system call interrupt

    mov eax, 1        ; opcode for exit system call
    mov ebx, 0        ; 1st arg, exit(0)
    int 0x80          ; system call interrupt

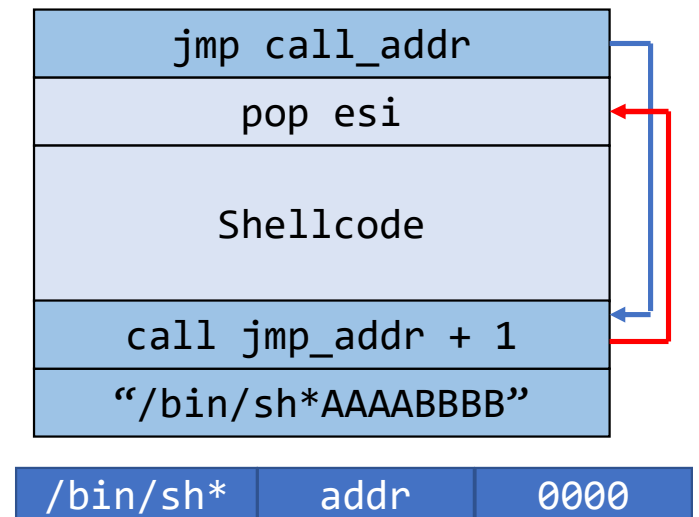
saveme:
    ??                ; (complete)
    msg db "Hello, world!", 0xA, 0xD
```

Activity 2: Spawn a new Shell

- A *working* startup code is provided that pushes data on stack, you need to:
 - Provide arguments to the spawned shell
 - Provide environment variables to the spawned shell

Activity 2: Spawn a new Shell

- A startup code is provided that uses relative addressing, you need to:
 - Complete the missing parts
 - Answer few questions
- You need to replace:
 - * with a NULL byte
 - AAAA with the address of the address of string
 - BBBB with NULL bytes
 - Why cannot we start with `/bin/sh0AAAA0000`?
- Can a program modify the code segment?
 - How can we solve this issue?



-
- `mov [ebx+7], 0x00`

<code>/bin/sh*</code>	<code>addr</code>	<code>0000</code>
-----------------------	-------------------	-------------------

Questions?
