Dealing with shellcode

Tooling ourselves to open that code

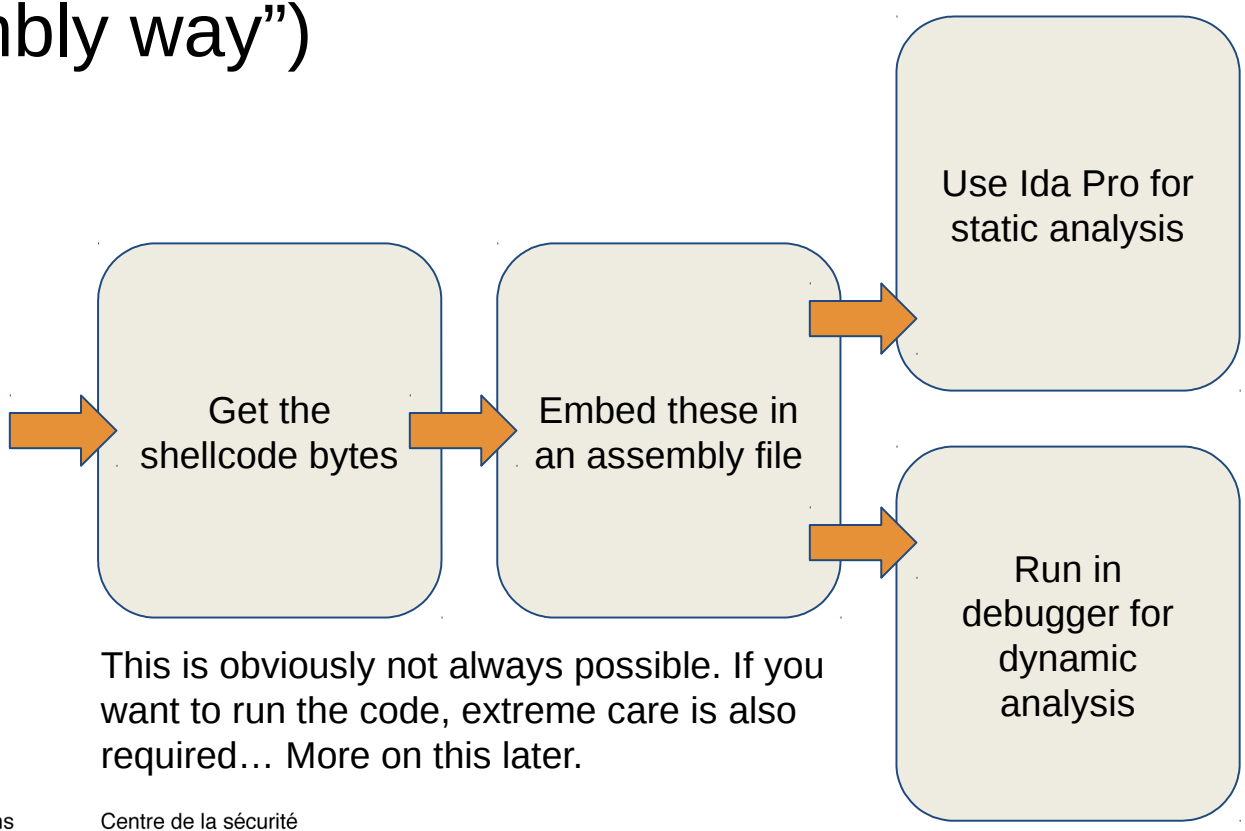
What is a shellcode?

Shellcode

- A shellcode is a piece of software that is usually present as part of an attack
  - As part of an exploit, the shellcode will generally be the code that will be launched after the attacker gained code execution on a machine
- Shellcodes tend to be small
- A great deal of care is put in crafting these
  - A bad shellcode would mean a failure for an attacker
  - Often need to remove all NULL chars from a shellcode
  - Need to be position independent code (PIC)
- Shellcode analysis can be a bit daunting if you never did it
  - This is why we're going to do a small break in the programming aspect of the class and will be doing shellcode analysis for a little while! :)

All shellcode analysed here are part of the open source Metasploit framework

Shellcode analysis process (easy “hey I know assembly way”)



This is obviously not always possible. If you want to run the code, extreme care is also required… More on this later.

The dynamic way

Let’s pretend the following bytes were found and are suspected part of a shell code:

```
6a 0b 58 99 52 66 68 2d 63 89 e7 68 2f 73 68
00 68 2f 62 69 6e 89 e3 52 e8 09 00 00 00 65
63 68 6f 20 6d 30 30 00 57 53 89 e1 cd 80
```

If you are familiar with x86 opcode, you might have noticed the ending:

```
0xCD 0x80
```

This generally suggest a system call since this directly translate to:

```
int 0x80
```

And therefore suggest a possible Linux shellcode

Original code used in this example can be found at please play fairgame and don’t check the original code before the end of demo::
https://github.com/rapid7/metasploit-framework/blob/master/external/source/shellcode/linux/ia32/single_exec.asm

The dynamic way

Why do the unconditional jump to main in a case like this one?

Packaging the shellcode like this has many advantages...

Also when you actually are ready to run the code, you can change “jmp main” to “jmp shellcode” and you will be able to run the shellcode in a debugger.

```
 2 section .text
 3 global main
 4
 5 main:
 6     jmp main
 7
 8 shellcode:
 9     db 0x6a, 0x0b, 0x58, 0x99, 0x52, 0x66, 0x68
10     db 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73
11     db 0x68, 0x00, 0x68, 0x2f, 0x62, 0x69, 0x6e
12     db 0x89, 0xe3, 0x52, 0xe8, 0x09, 0x00, 0x00
13     db 0x00, 0x65, 0x63, 0x68, 0x6f, 0x20, 0x6d
14     db 0x30, 0x30, 0x00, 0x57, 0x53, 0x89, 0xe1
15     db 0xcd, 0x80
```

Original code used in this example can be found at please play fairgame and don’t check the original code before the end of demo::
https://github.com/rapid7/metasploit-framework/blob/master/external/source/shellcode/linux/ia32/single_exec.asm

Let’s do this one together and then you’ll do one on your own.