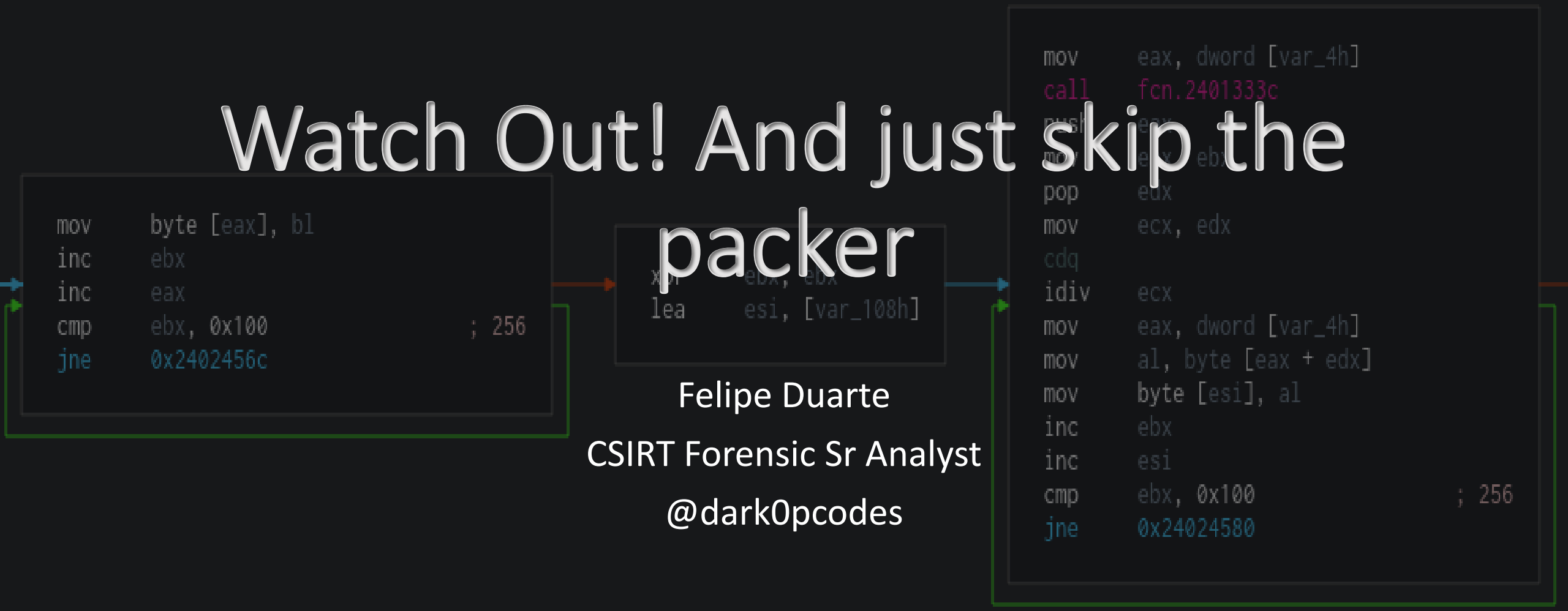
Watch Out! And just skip the packer

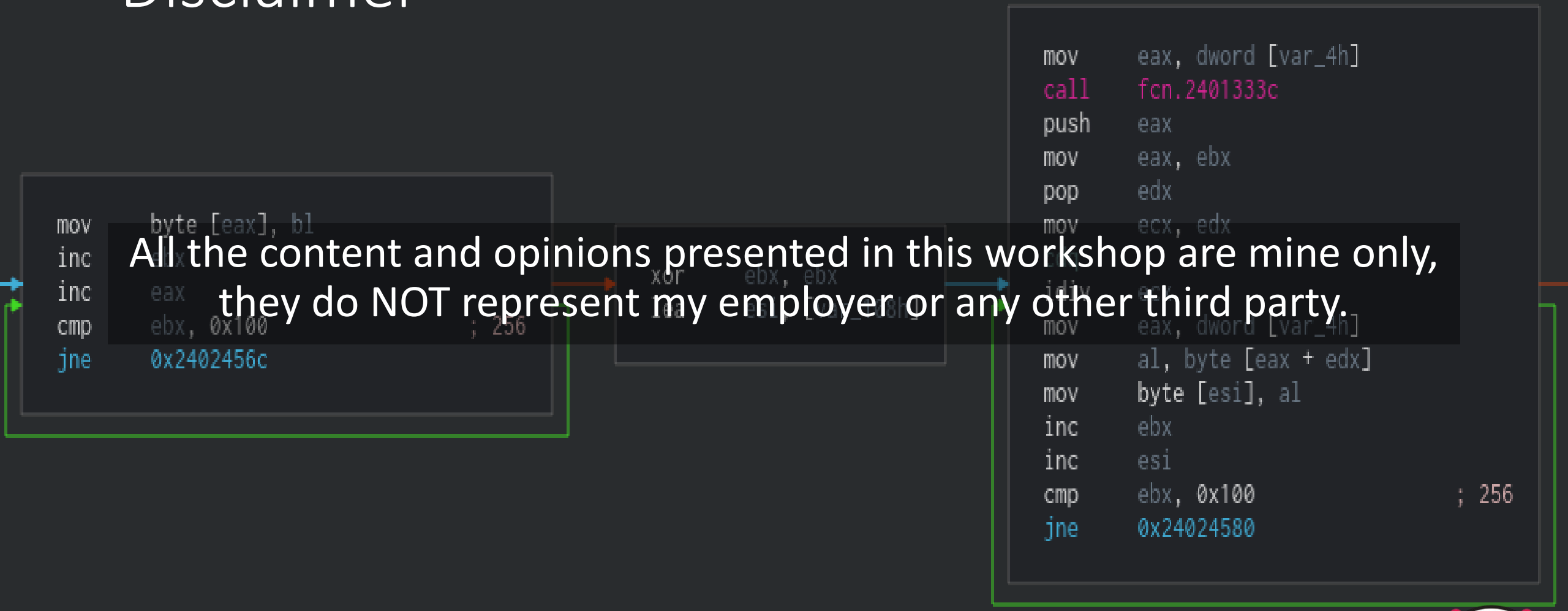


Felipe Duarte

CSIRT Forensic Sr Analyst

@dark0pcodes

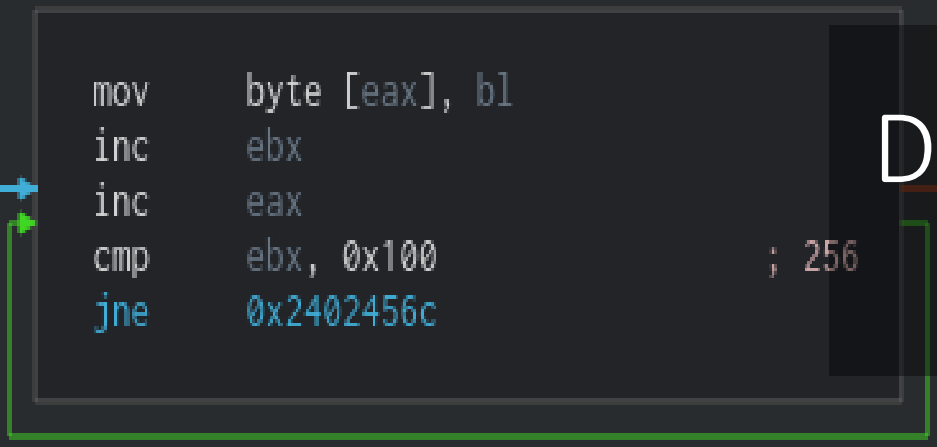
Disclaimer



All the content and opinions presented in this workshop are mine only, they do NOT represent my employer or any other third party.

Divide et impera

Julius Caesar



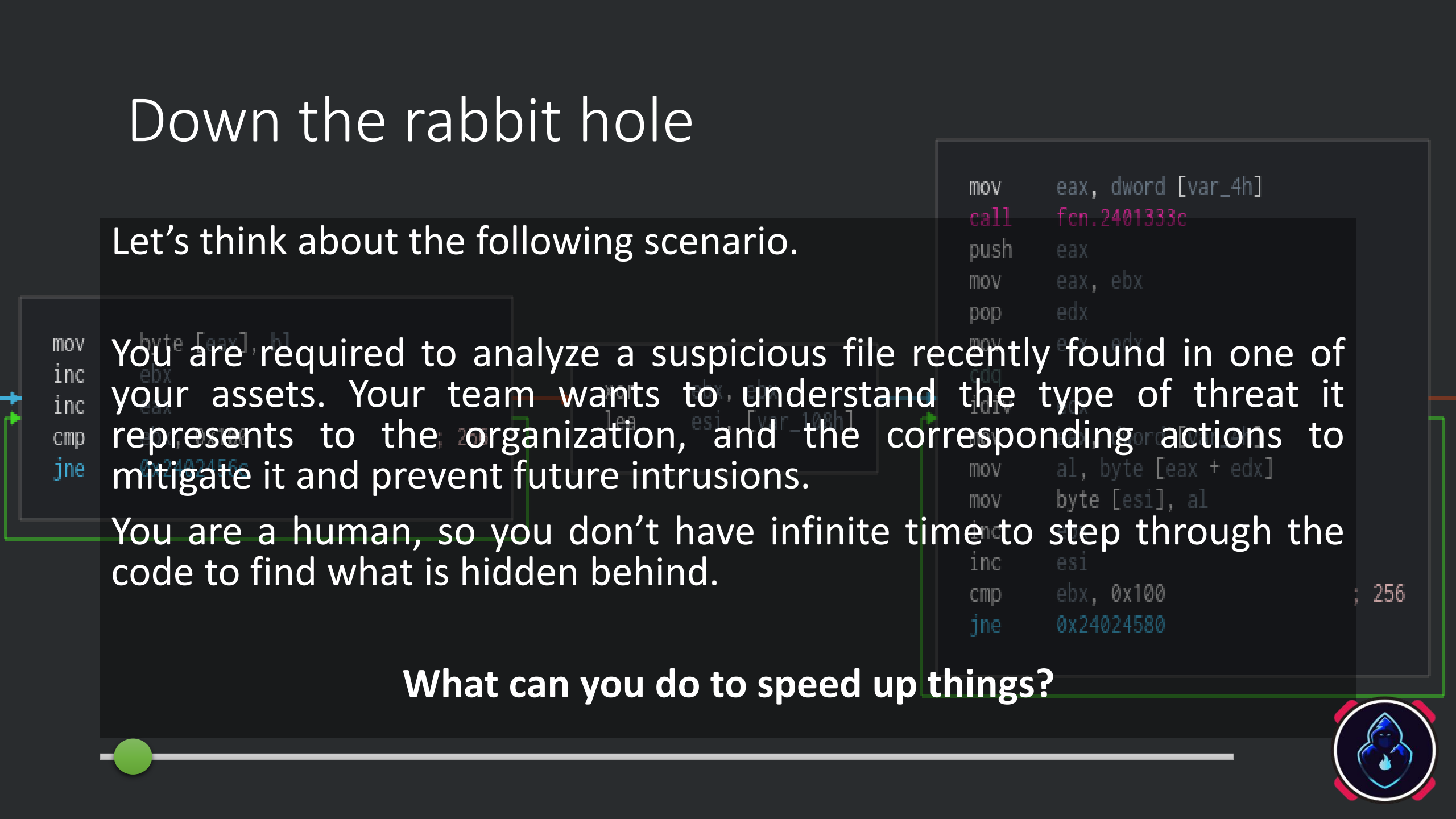
Down the rabbit hole

Let’s think about the following scenario.

You are required to analyze a suspicious file recently found in one of your assets. Your team wants to understand the type of threat it represents to the organization, and the corresponding actions to mitigate it and prevent future intrusions.

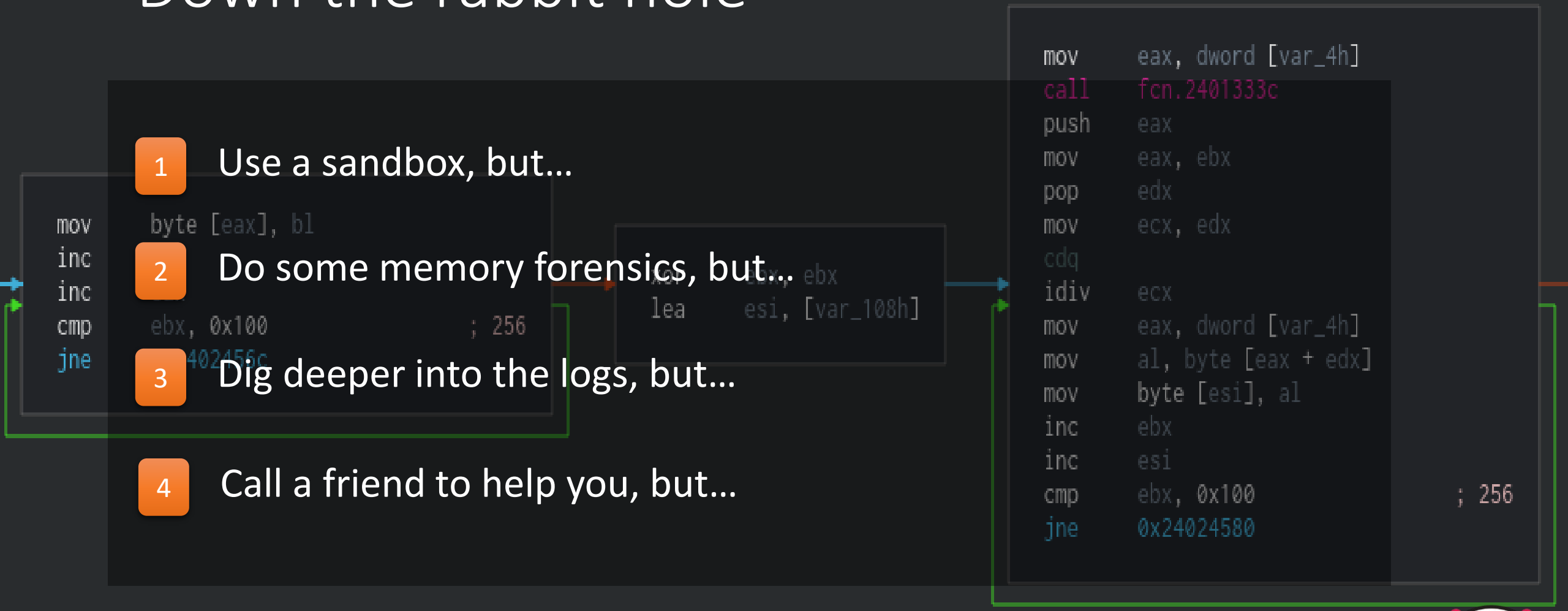
You are a human, so you don’t have infinite time to step through the code to find what is hidden behind.

What can you do to speed up things?

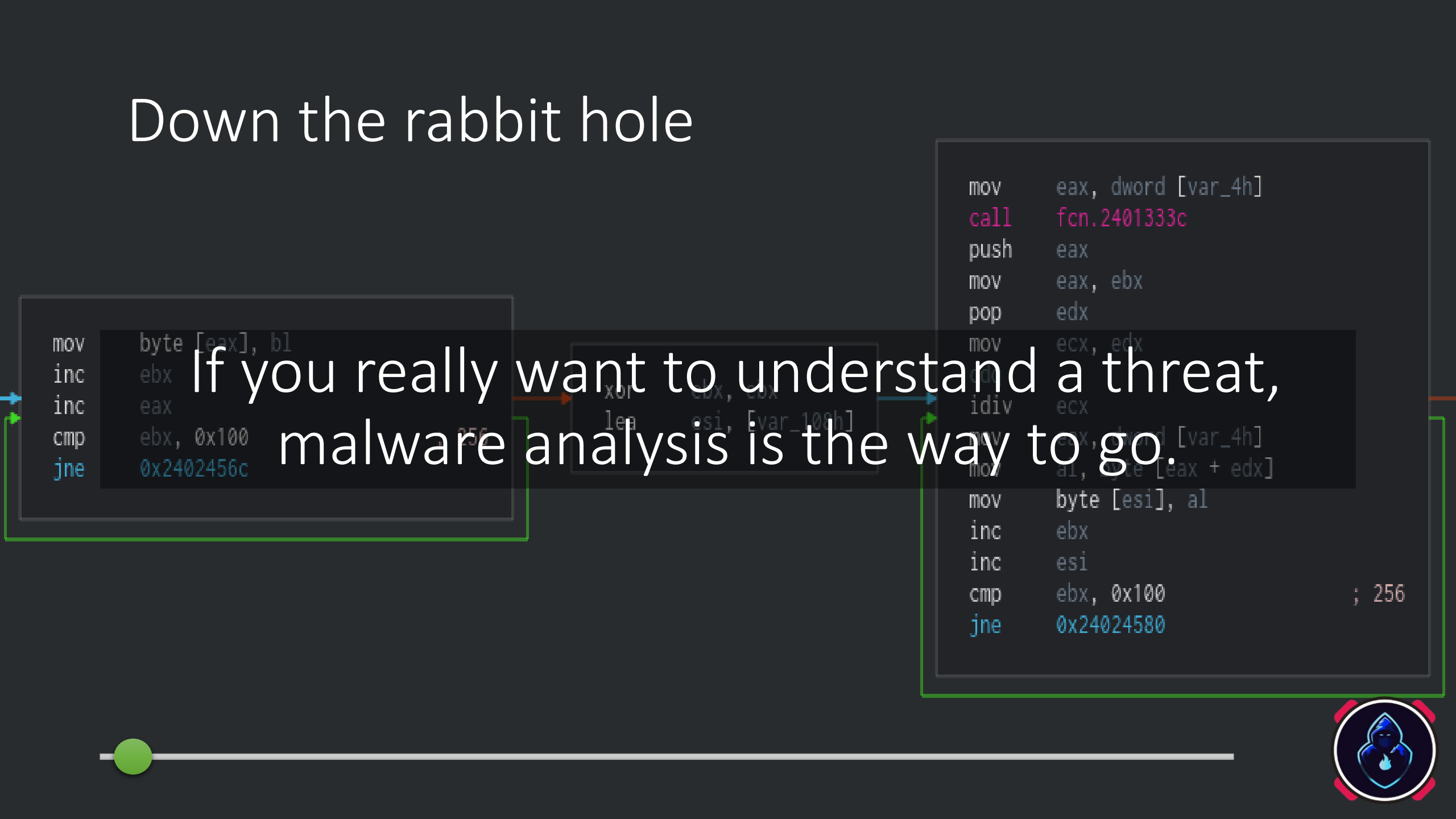


Down the rabbit hole

1. Use a sandbox, but…
2. Do some memory forensics, but…
3. Dig deeper into the logs, but…
4. Call a friend to help you, but…

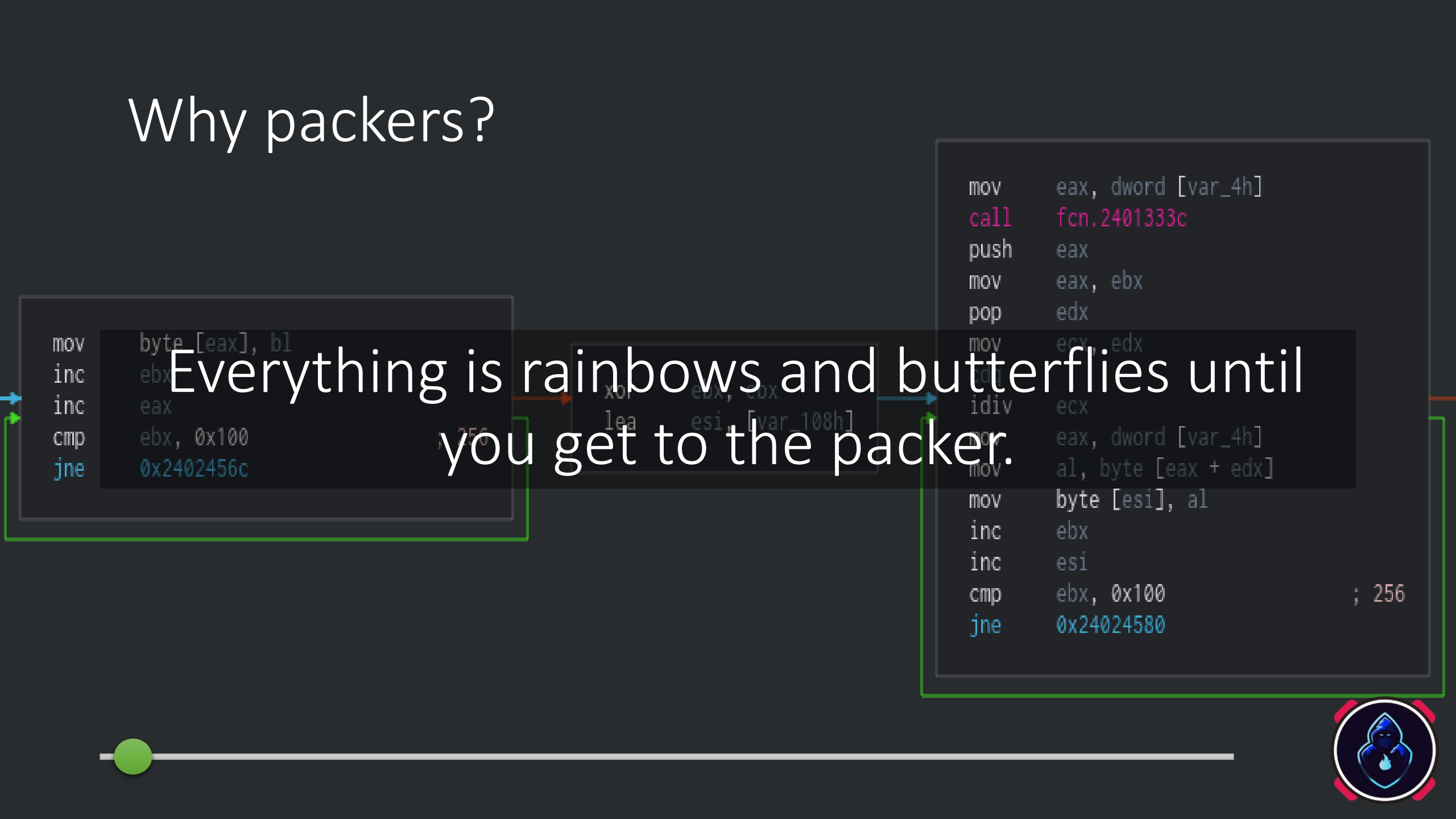


Down the rabbit hole



If you really want to understand a threat, malware analysis is the way to go.

Why packers?



Everything is rainbows and butterflies until you get to the packer.

What is a packer?

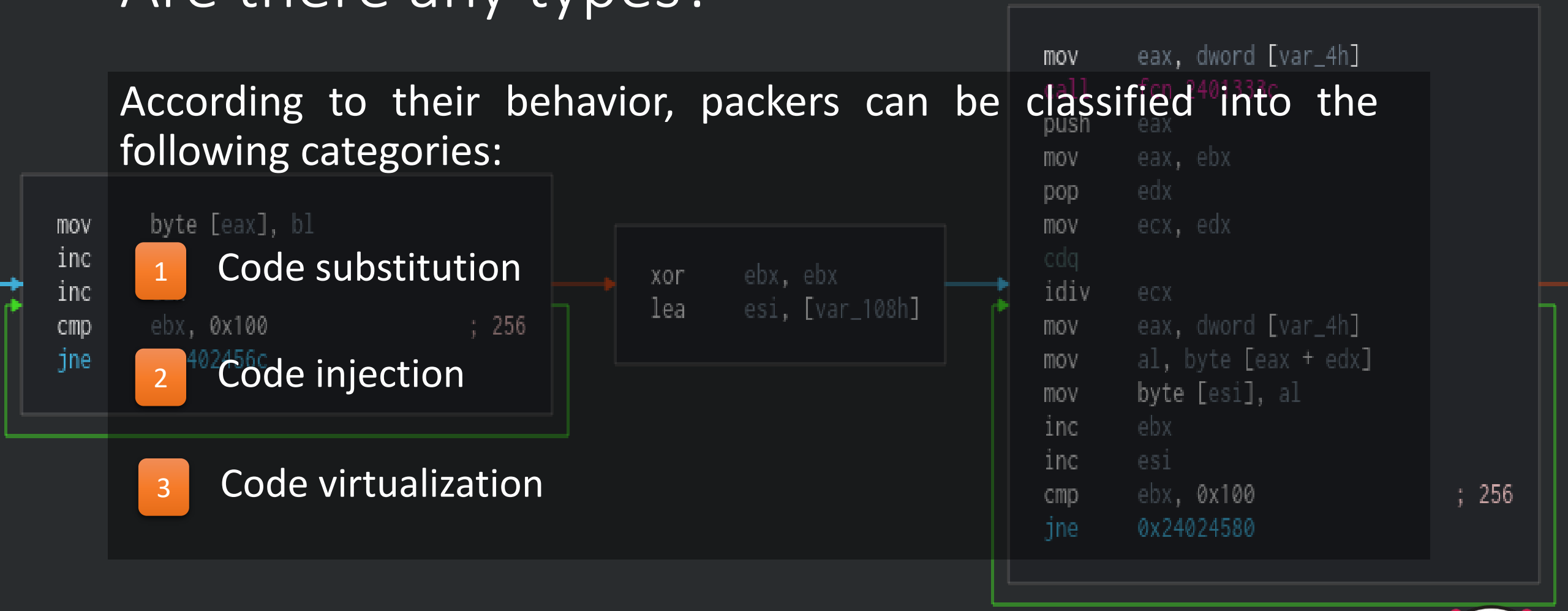
Tool used by software developers (malicious or not) to shield programs against reverse engineering. They can provide protection at different levels such as:

1. Anti-analysis checks
2. Signature avoidance
3. Increase the software complexity

Are there any types?

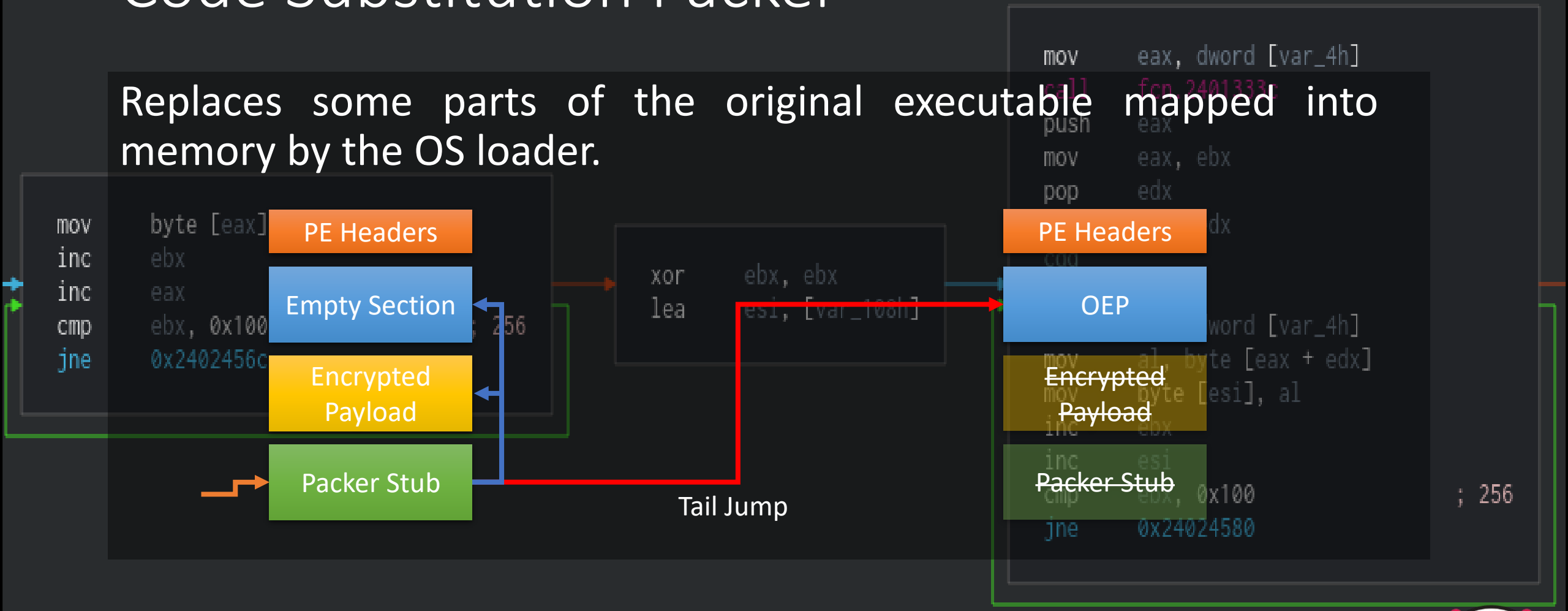
According to their behavior, packers can be classified into the following categories:

1. Code substitution
2. Code injection
3. Code virtualization



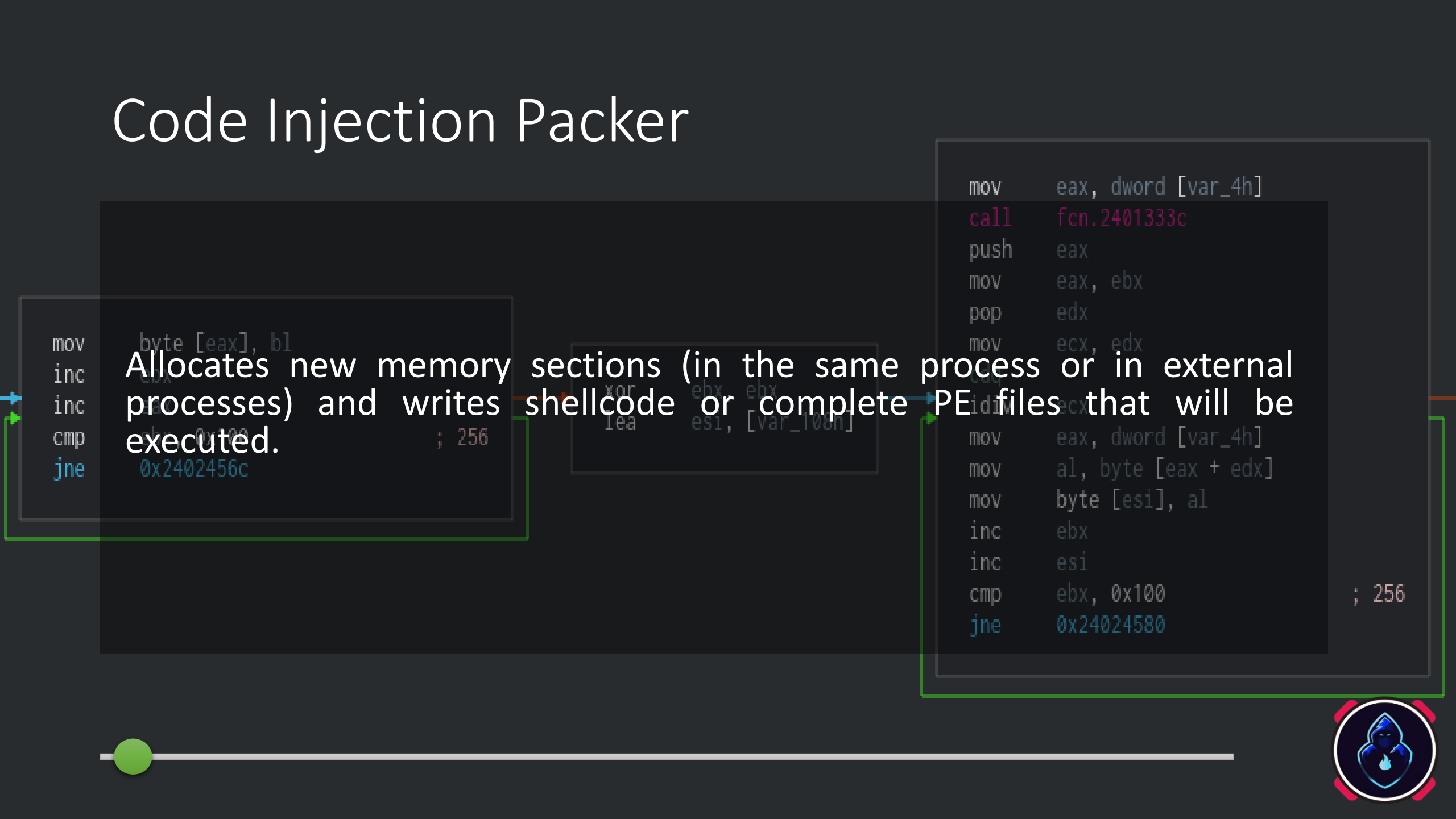
Code Substitution Packer

Replaces some parts of the original executable mapped into memory by the OS loader.

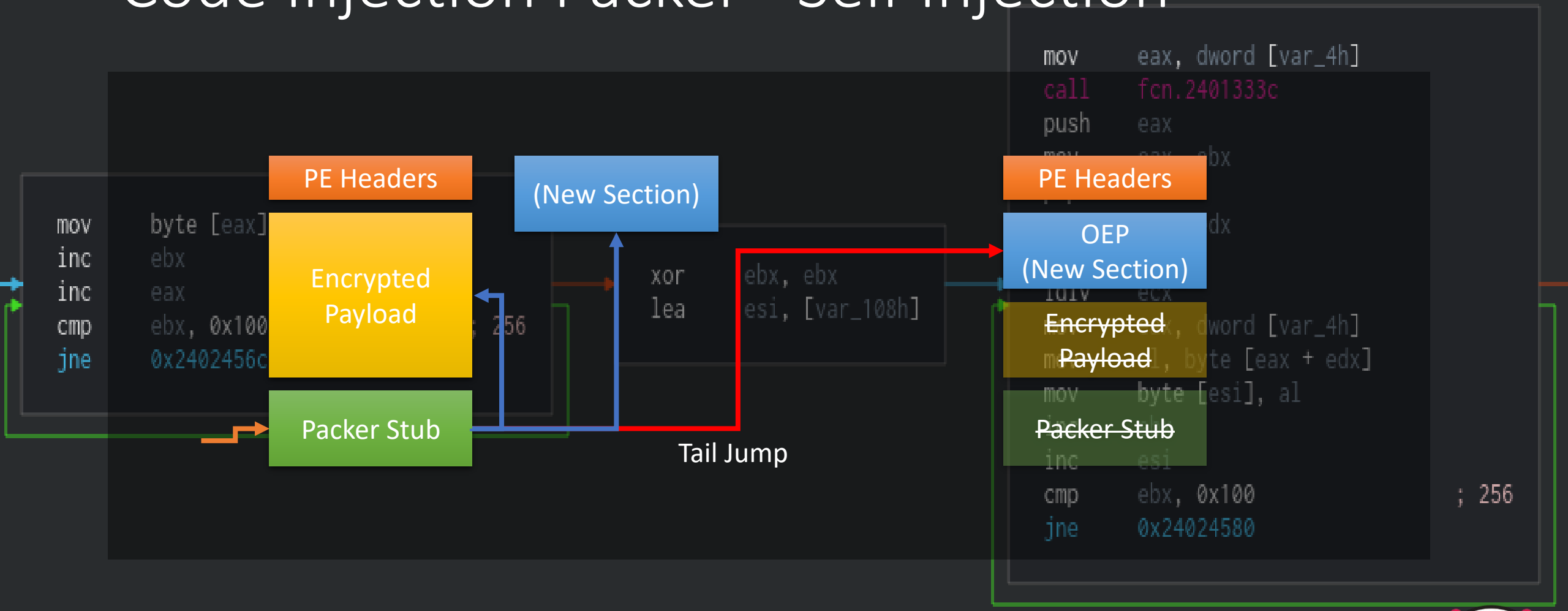


Code Injection Packer

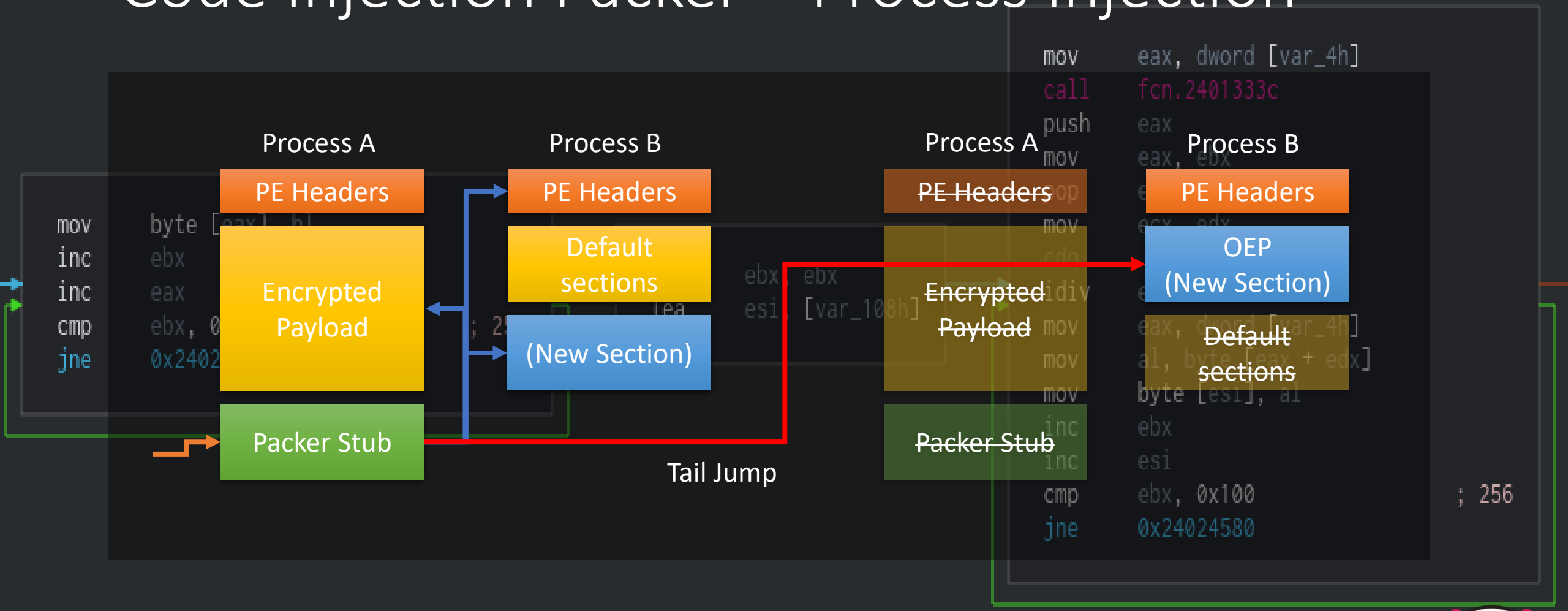
Allocates new memory sections (in the same process or in external processes) and writes shellcode or complete PE files that will be executed.



Code Injection Packer - Self Injection

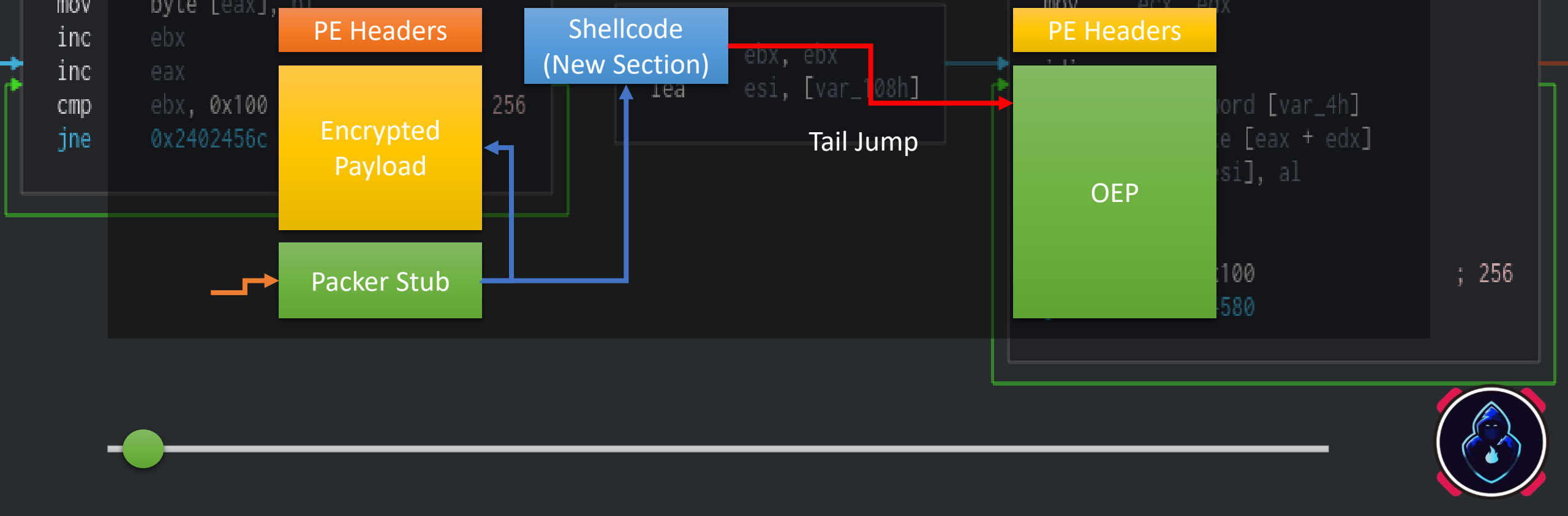


Code Injection Packer – Process Injection



Hybrid Packer - Self Injection & Substitution

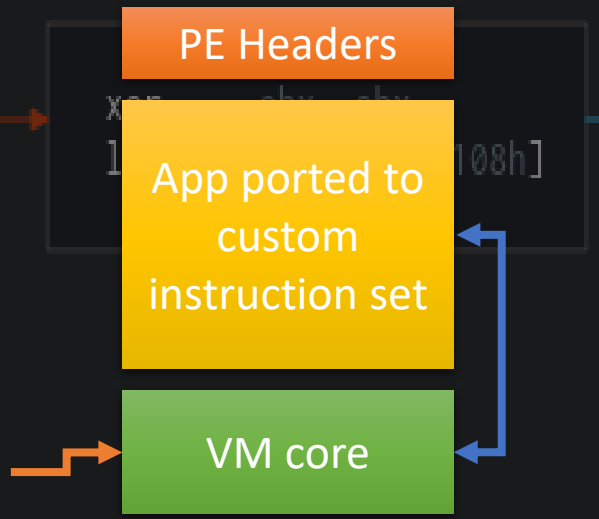
Allocates new memory sections in the same process and writes intermediate shellcode that replaces the original executable mapped into memory by the OS loader.



Code Virtualization Packer

Contains a virtual machine and a copy of the program ported to a custom set of instructions (only known by the VM) that are interpreted in run-time.

```
mov     byte [eax], bl
inc     ebx
inc     eax
cmp     ebx, 0x100                      ; 256
jne     0x2402456c
```




```
mov     eax, dword [var_4h]
call    fcn.24013334
push    eax
mov     eax, ebx
pop     edx
mov     ecx, edx
cdq
idiv    ecx
mov     eax, dword [var_4h]
mov     al, byte [eax + edx]
mov     byte [esi], al
inc     ebx
inc     esi
cmp     ebx, 0x100                      ; 256
jne     0x24024580
```

```
mov     byte [eax], bl
inc     ebx
inc     eax
cmp     ebx, 0x100                ; 256
jne     0x2402456c
```

```
xor     ebx, ebx
lea     ecx, [var_1ch]
```

```
mov     eax, dword [var_4h]
call    fcn.2401333c
push    eax
mov     eax, ebx
pop     edx
mov     ecx, edx
cdq
idiv    ecx
mov     eax, dword [var_4h]
mov     al, byte [eax + edx]
mov     byte [esi], al
inc     ebx
inc     esi
cmp     ebx, 0x100                ; 256
jne     0x24024580
```

Let’s dig into the details!