black hat EUROPE 2017

DECEMBER 4-7, 2017
EXCEL / LONDON, UK

POLITECNICO MILANO 1863

DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

Hiding PIN's Artifacts to Defeat Evasive Malware

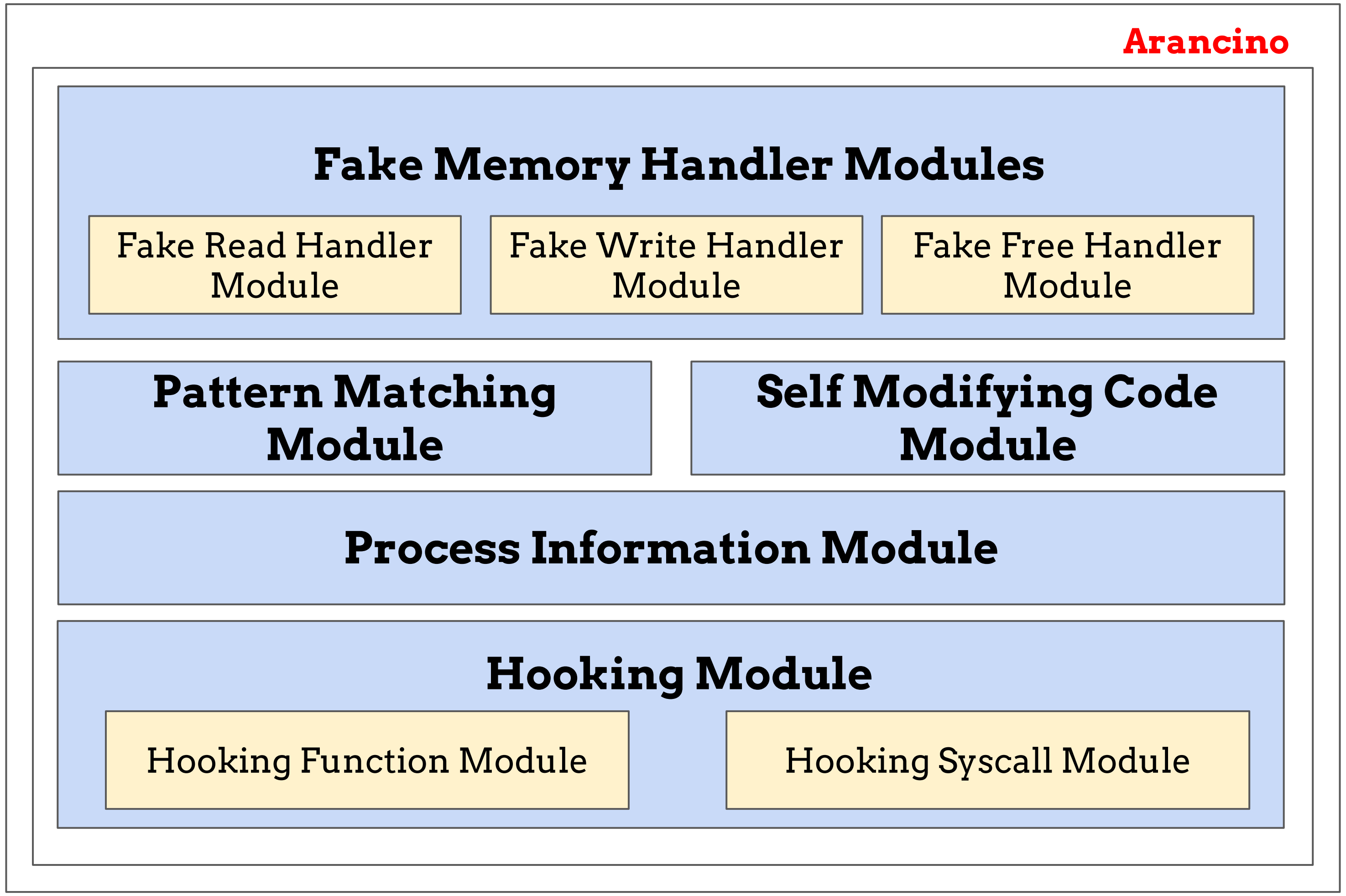
Mario Polino, Andrea Continella, Sebastiano Mariani, Lorenzo Fontana, Stefano D'Alessio, Fabio Gritti, Stefano Zanero

Agenda

- Arancino
- Dynamic Binary Instrumentation Tools
- DBI Evasion
- Evasive Malware Measurement
- Evasive Resilient Unpacking Tool
- DEMO

Arancino

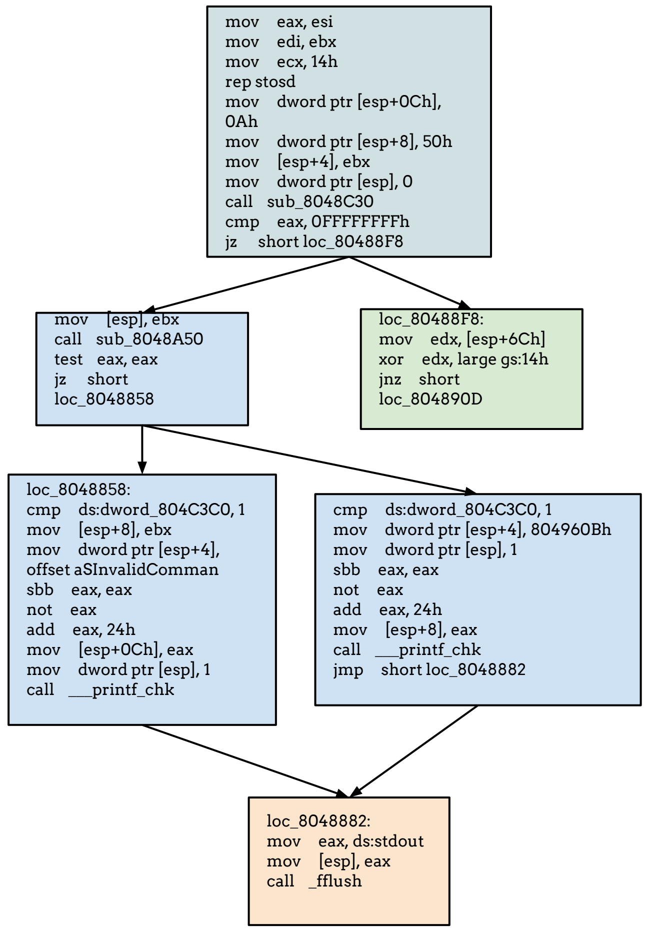
Arancino


Arancino
Fake Memory Handler Modules
Fake Read Handler Module
Fake Write Handler Module
Fake Free Handler Module
Pattern Matching Module
Self Modifying Code Module
Process Information Module
Hooking Module
Hooking Function Module
Hooking Syscall Module

Malware Analysis

Malware Analysis

Static



```
mov   eax, esi
mov   edi, ebx
mov   ecx, 14h
rep stosd
mov   dword ptr [esp+0Ch],
0Ah
mov   dword ptr [esp+8], 50h
mov   [esp+4], ebx
mov   dword ptr [esp], 0
call  sub_8048C30
cmp   eax, 0FFFFFFFFh
jz    short loc_80488F8
```

```
mov   [esp], ebx
call  sub_8048A50
test  eax, eax
jz    short
loc_8048858
```

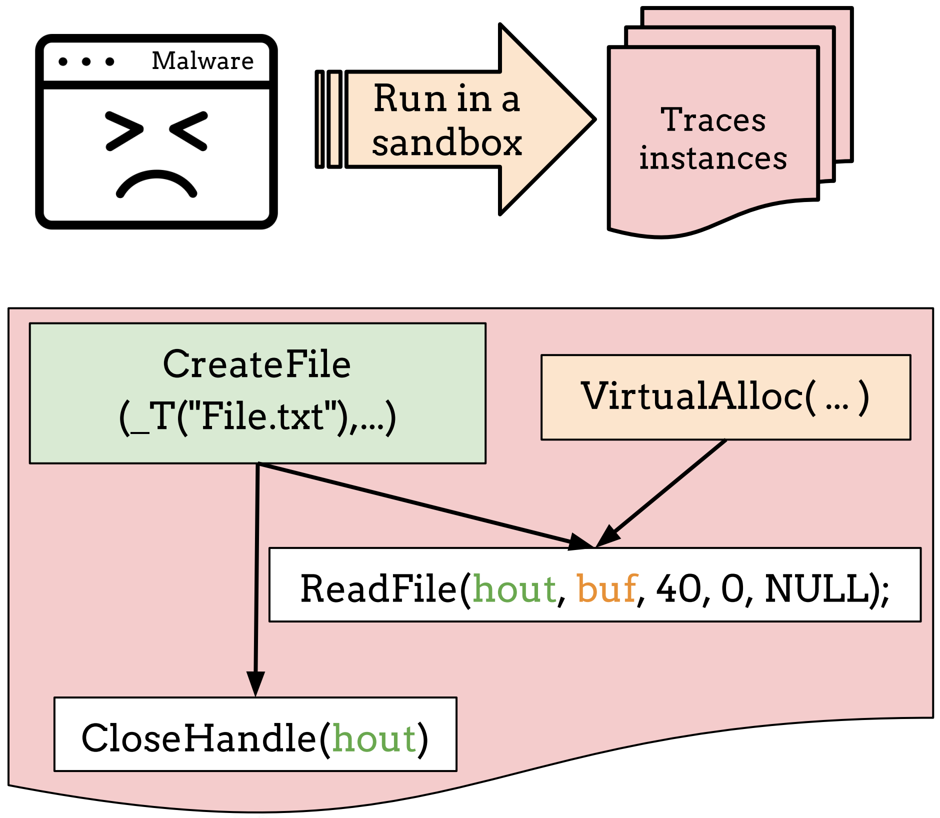
```
loc_80488F8:
mov   edx, [esp+6Ch]
xor   edx, large gs:14h
jnz   short
loc_804890D
```

```
loc_8048858:
cmp   ds:dword_804C3C0, 1
mov   [esp+8], ebx
mov   dword ptr [esp+4],
offset aSInvalidComman
sbb   eax, eax
not   eax
add   eax, 24h
mov   [esp+0Ch], eax
mov   dword ptr [esp], 1
call  ___printf_chk
```

```
cmp   ds:dword_804C3C0, 1
mov   dword ptr [esp+4], 804960Bh
mov   dword ptr [esp], 1
sbb   eax, eax
not   eax
add   eax, 24h
mov   [esp+8], eax
call  ___printf_chk
jmp   short loc_8048882
```

```
loc_8048882:
mov   eax, ds:stdout
mov   [esp], eax
call  _fflush
```

Dynamic

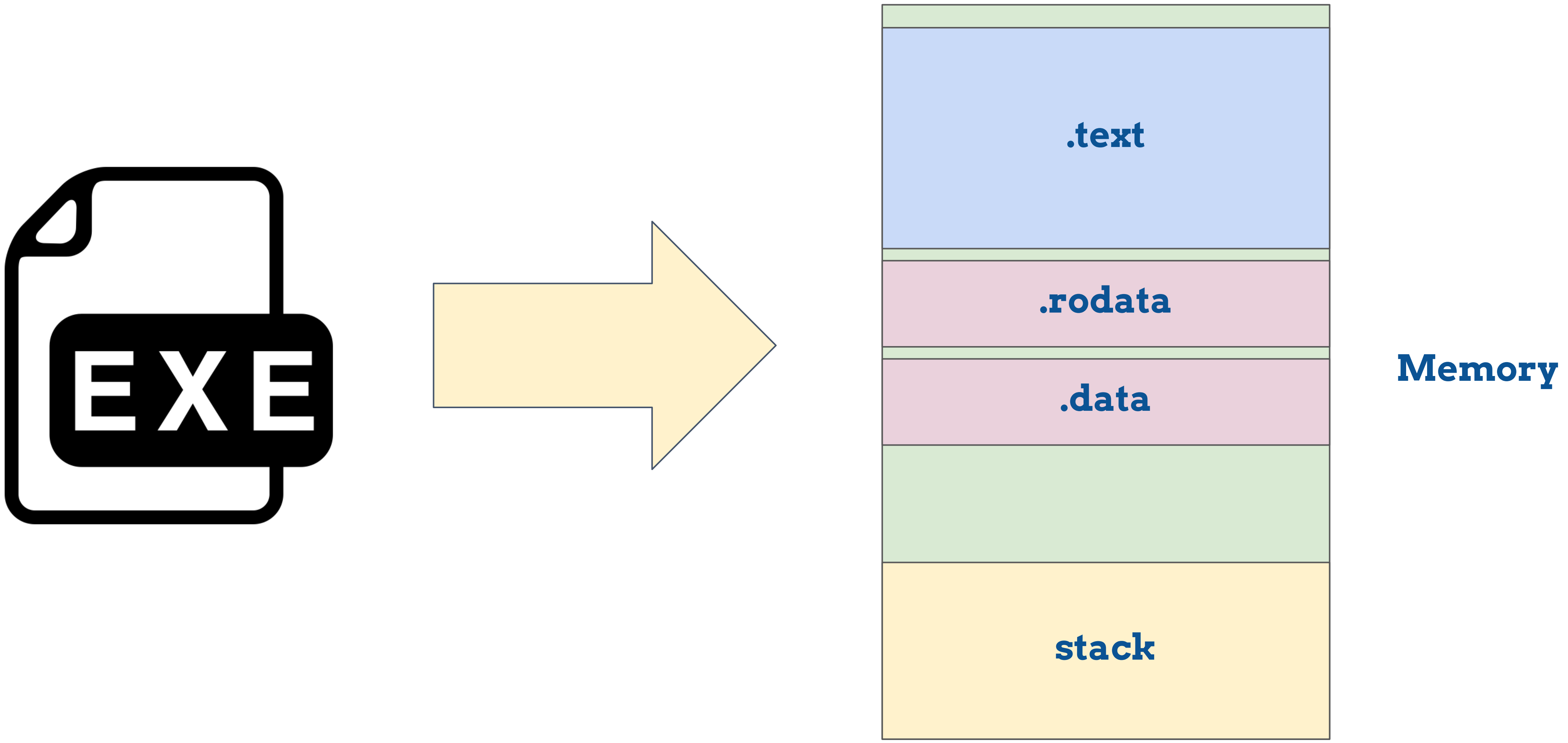


Malware Evasive

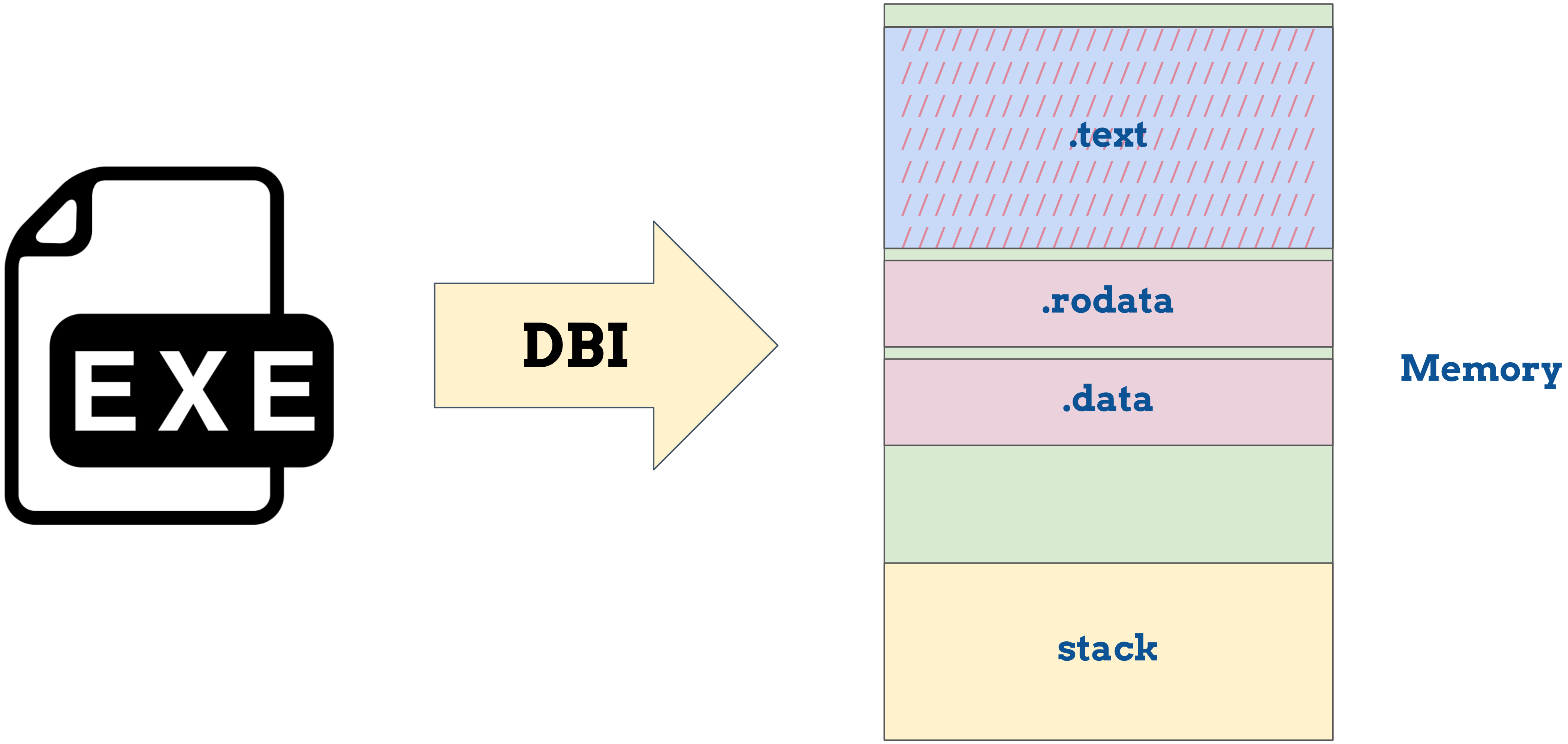
```
If (amIUnderAnalysis())
{
    die();
}
else
{
  beMalicious();
}
```

Dynamic Binary Instrumentation

What is a DBI Tool?



What is a DBI Tool?

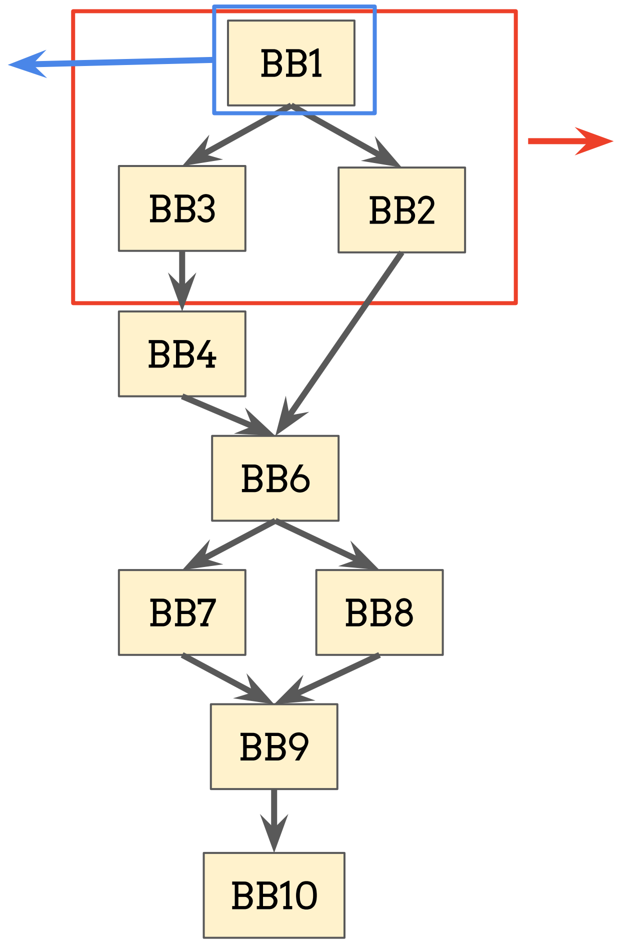


What is a DBI Tool?

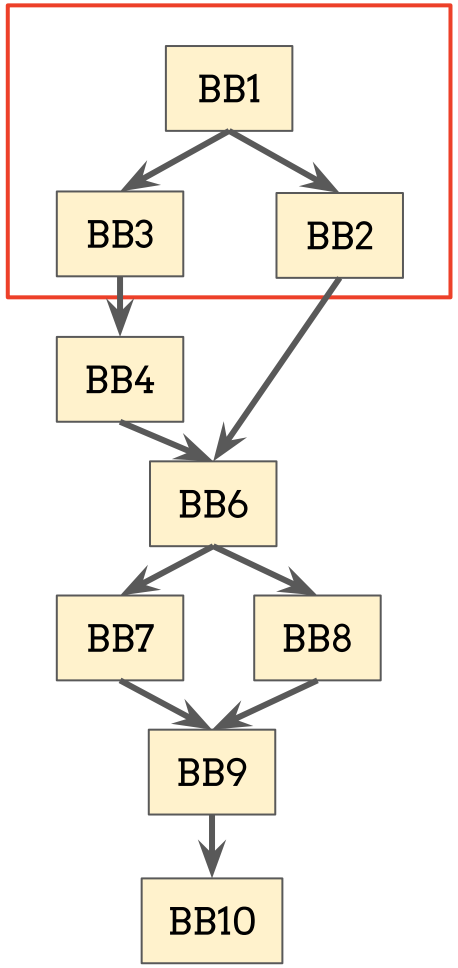
Basic Block

Trace

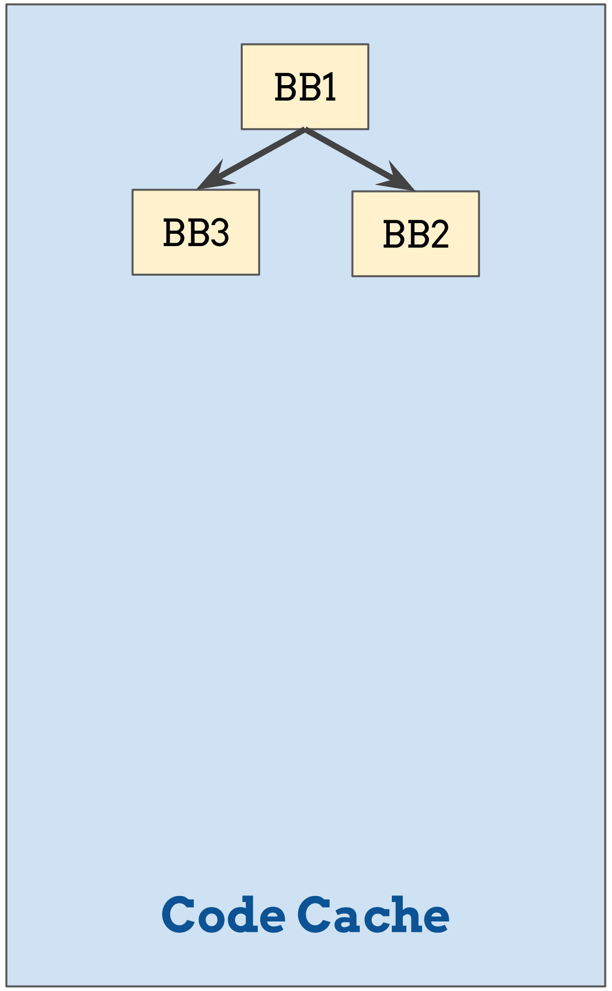
Control Flow Graph



What is a DBI Tool?

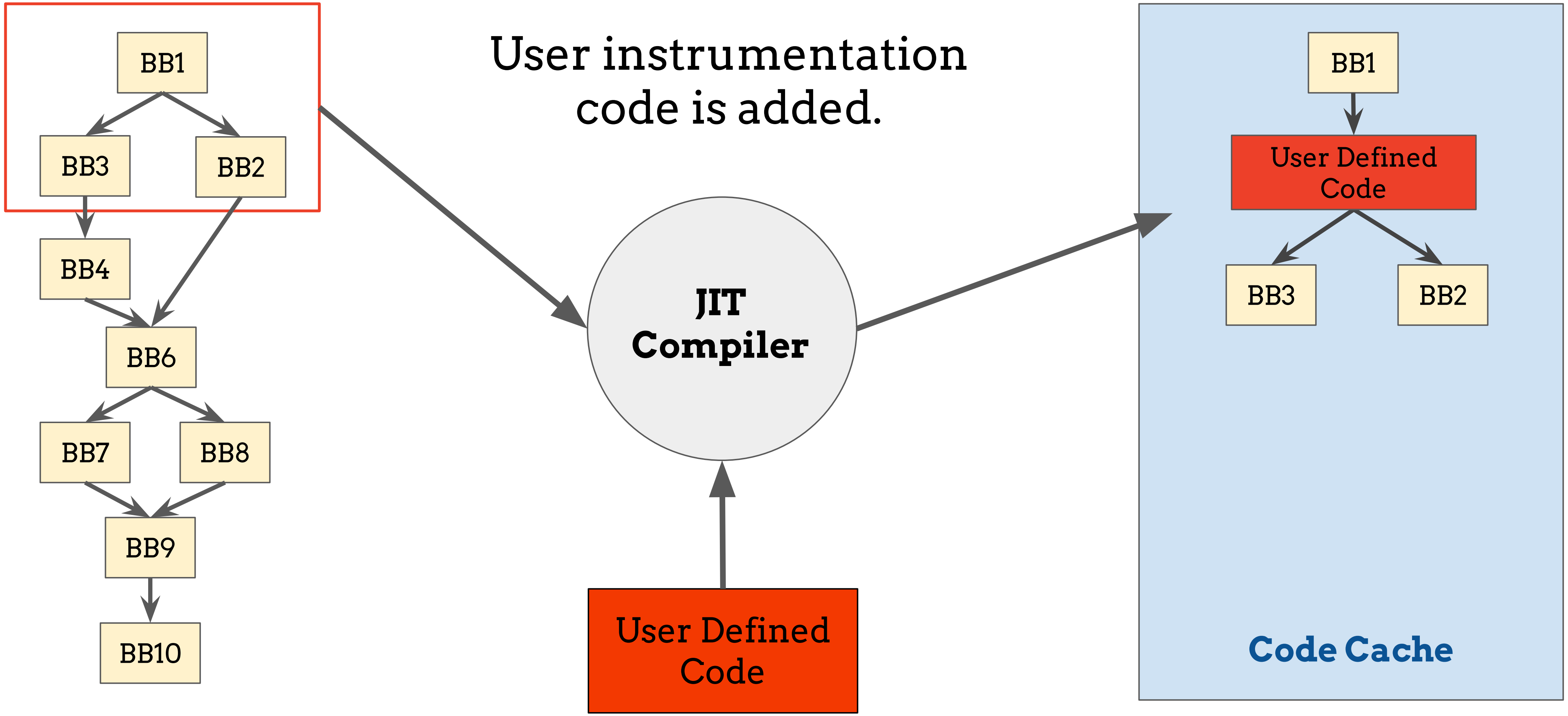


Trace is copied in the code cache



What is a DBI Tool?

User instrumentation code is added.



DBI - Evasive Malware

Pin

Intel Pin Tools

Dynamo RIO

The DR. is in.

DynamoRIO

Valgrind

rev.ng

rev.ng

DBI - Evasive Malware

Intel Pin Tools

DynamoRIO

Valgrind

rev.ng

DBI - Evasive Malware

Code Cache Artifacts

JIT Compiler Detection

Environment Artifact

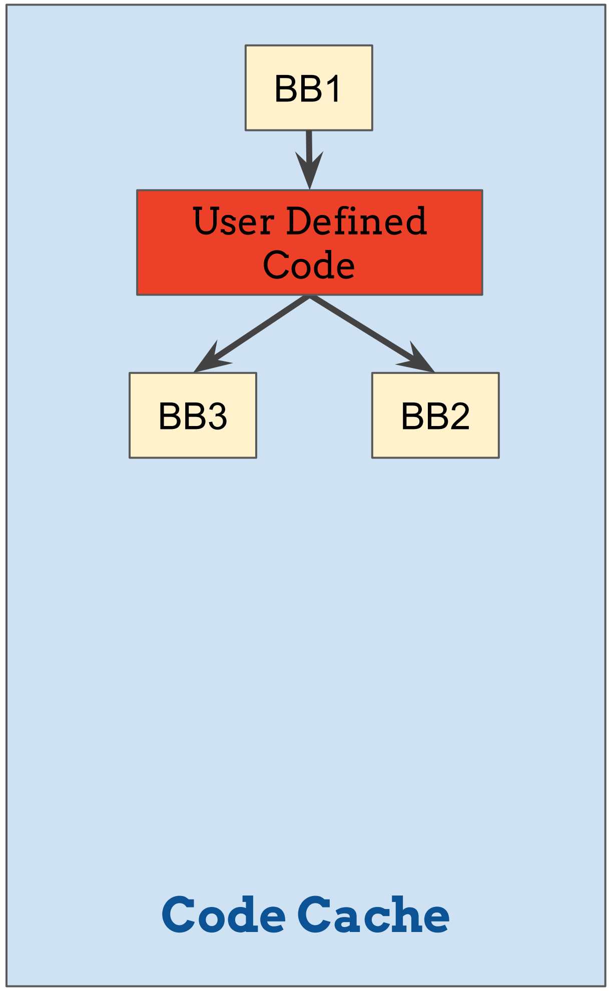
Overhead Detection

Code Cache Artifacts

Code Cache Artifacts

All those artifacts caused by having a Code Cache

- **IP Detection**
- **Self-Modifying Code**



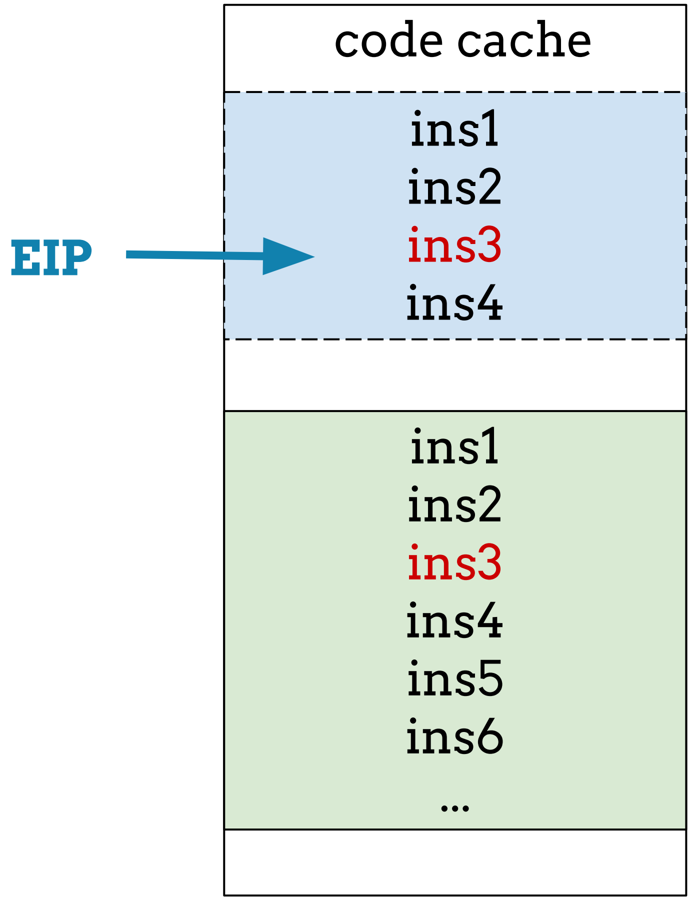
CCA - IP Detection

Nt Sycall (EIP -> EDX)

```
int 2e
```

Floating Point Context on the Stack

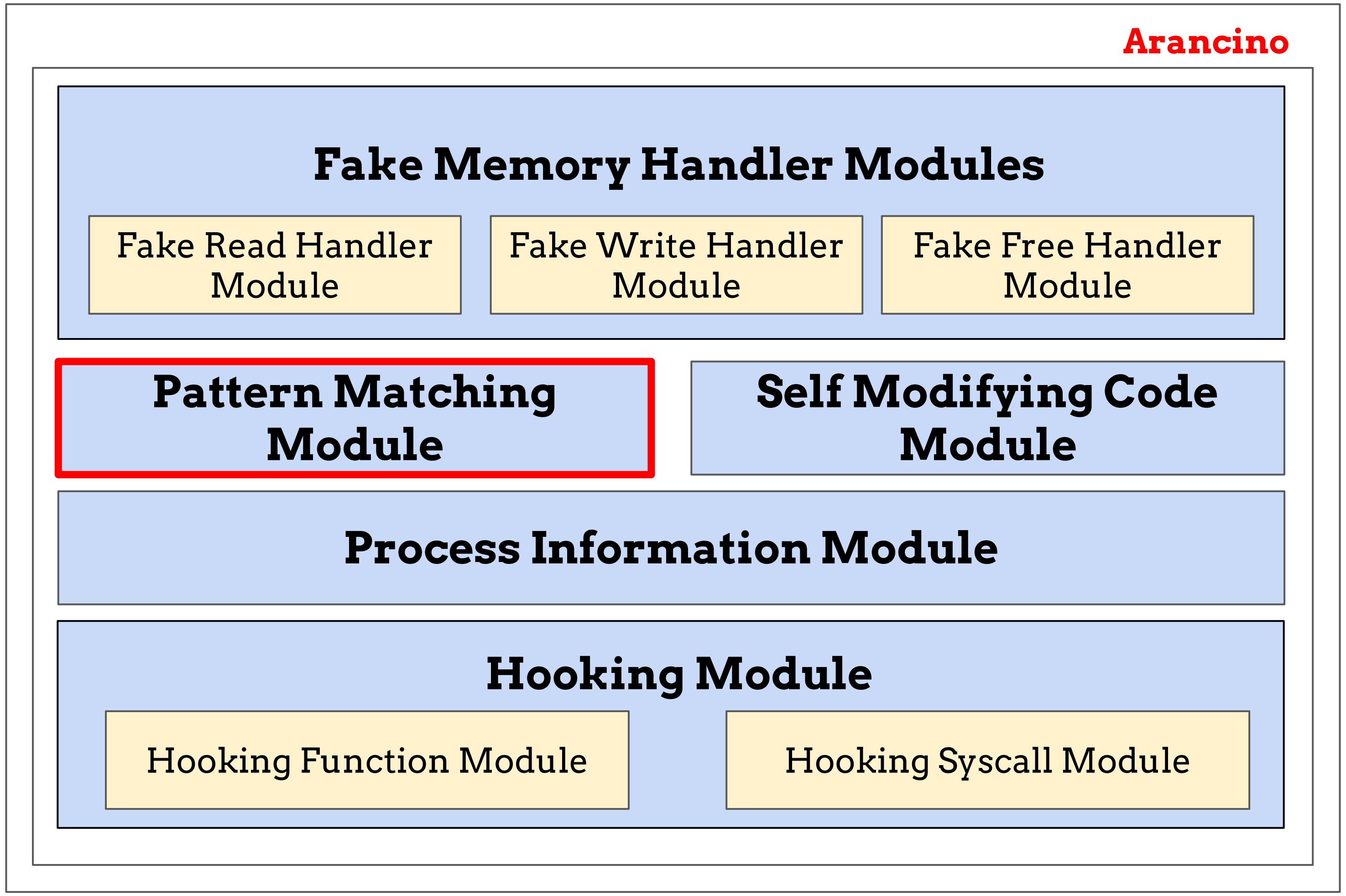
```
fsave/ fxsave/ fstenv
```



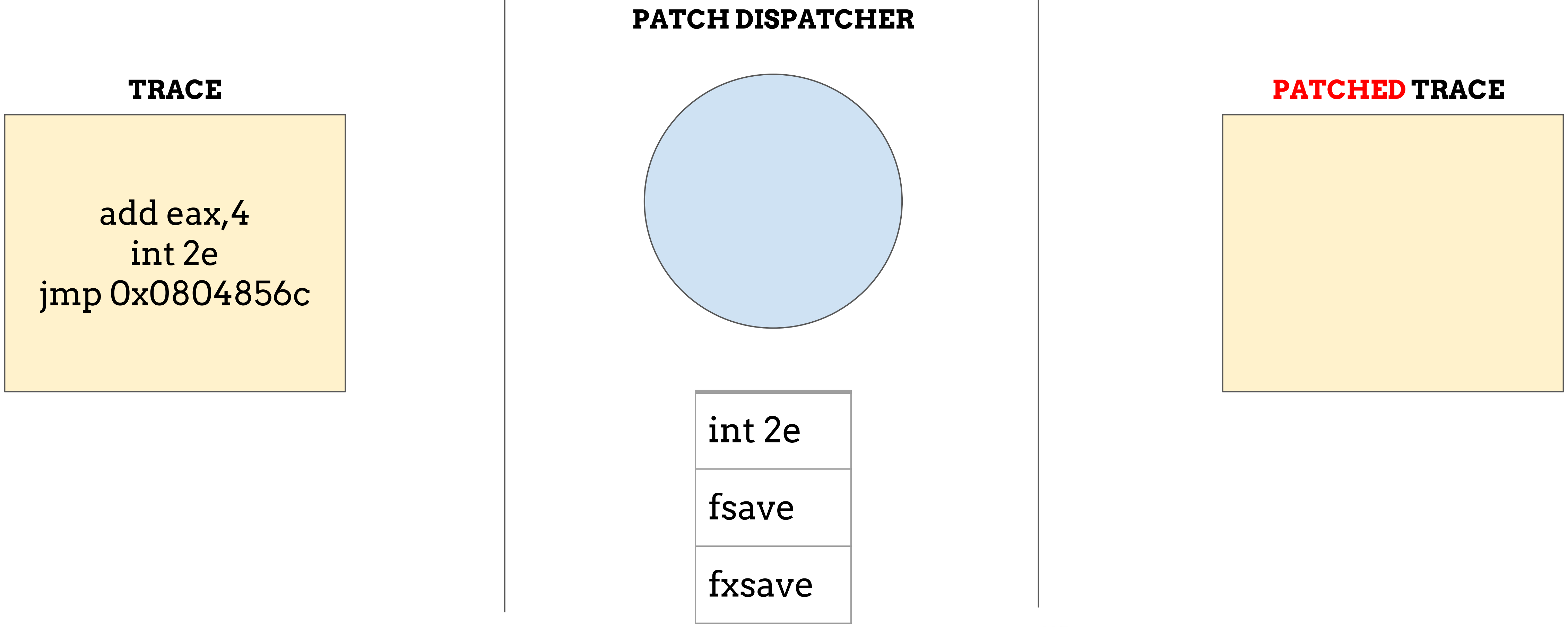
Arancino - Pattern Matching Module

- **PatchMap**: List of instructions and func pointers

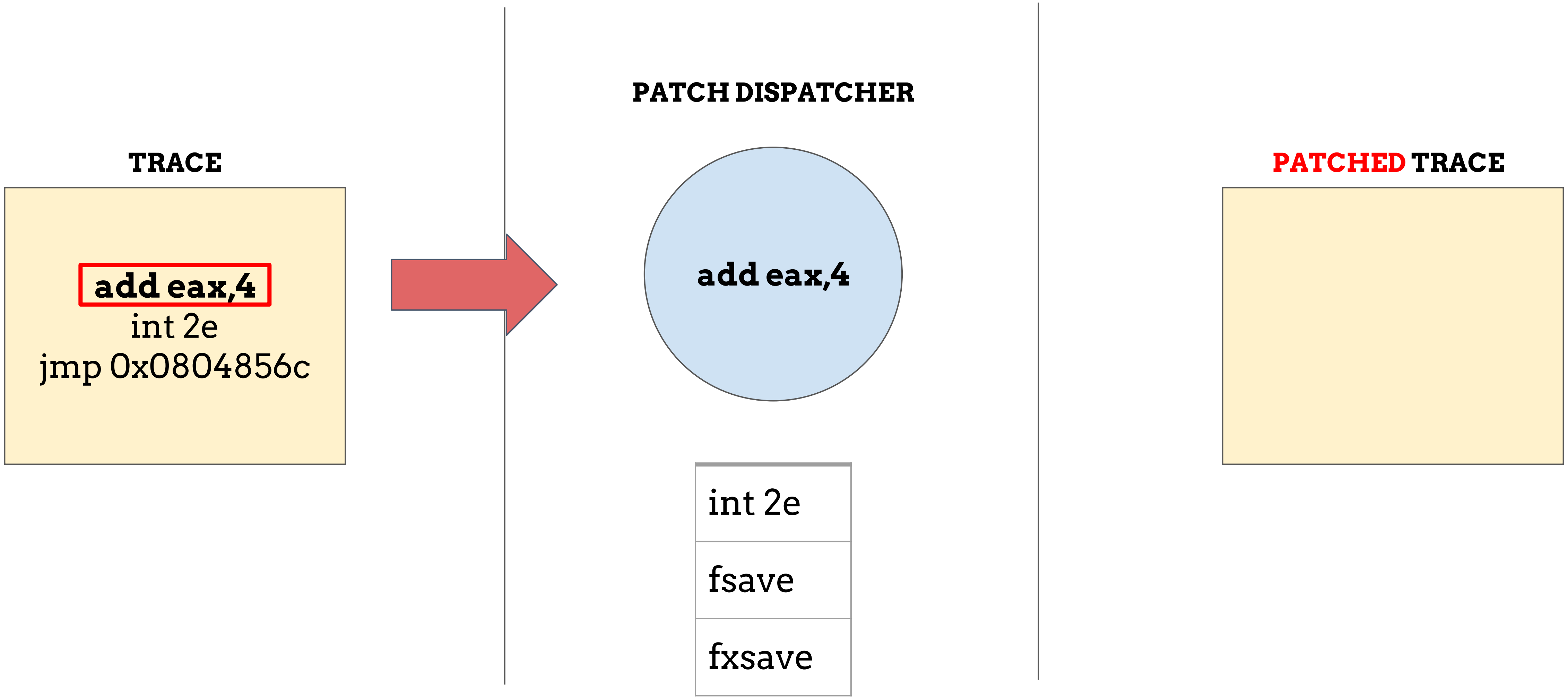
- **PatchDispatcher**: check and add patch to instructions during trace building.



CCA - IP Detection



CCA - IP Detection

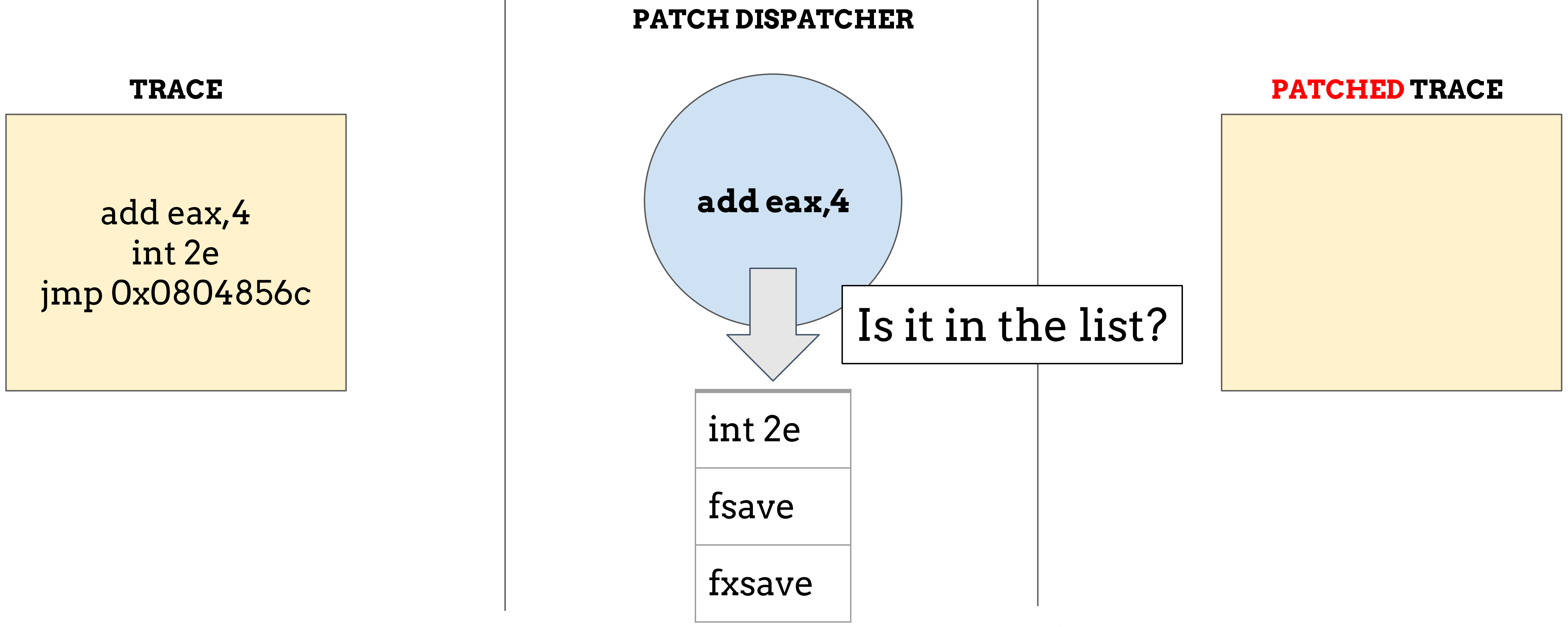


```
add eax,4
int 2e
jmp 0x0804856c
```

```
add eax,4
```

```
int 2e
fsave
fxsave
```

CCA - IP Detection



```
add eax,4
int 2e
jmp 0x0804856c
```

```
add eax,4
```

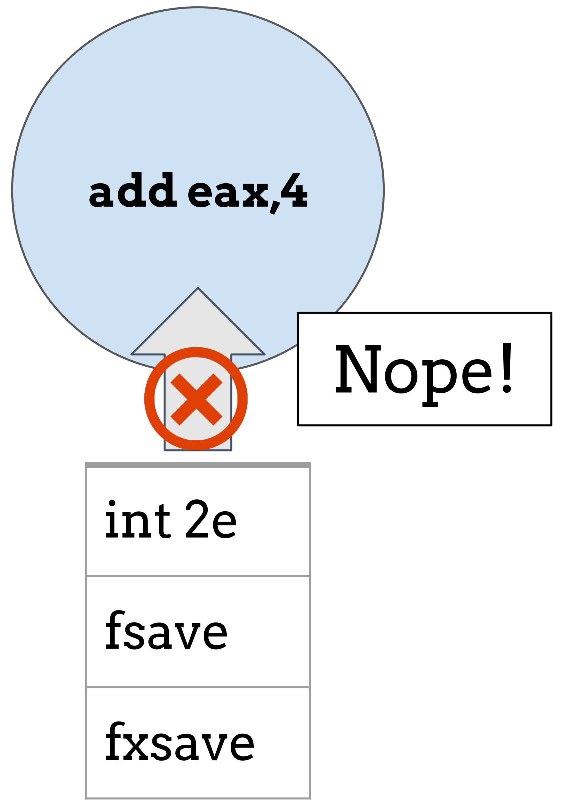
int 2e
fsave
fxsave

CCA - IP Detection

TRACE

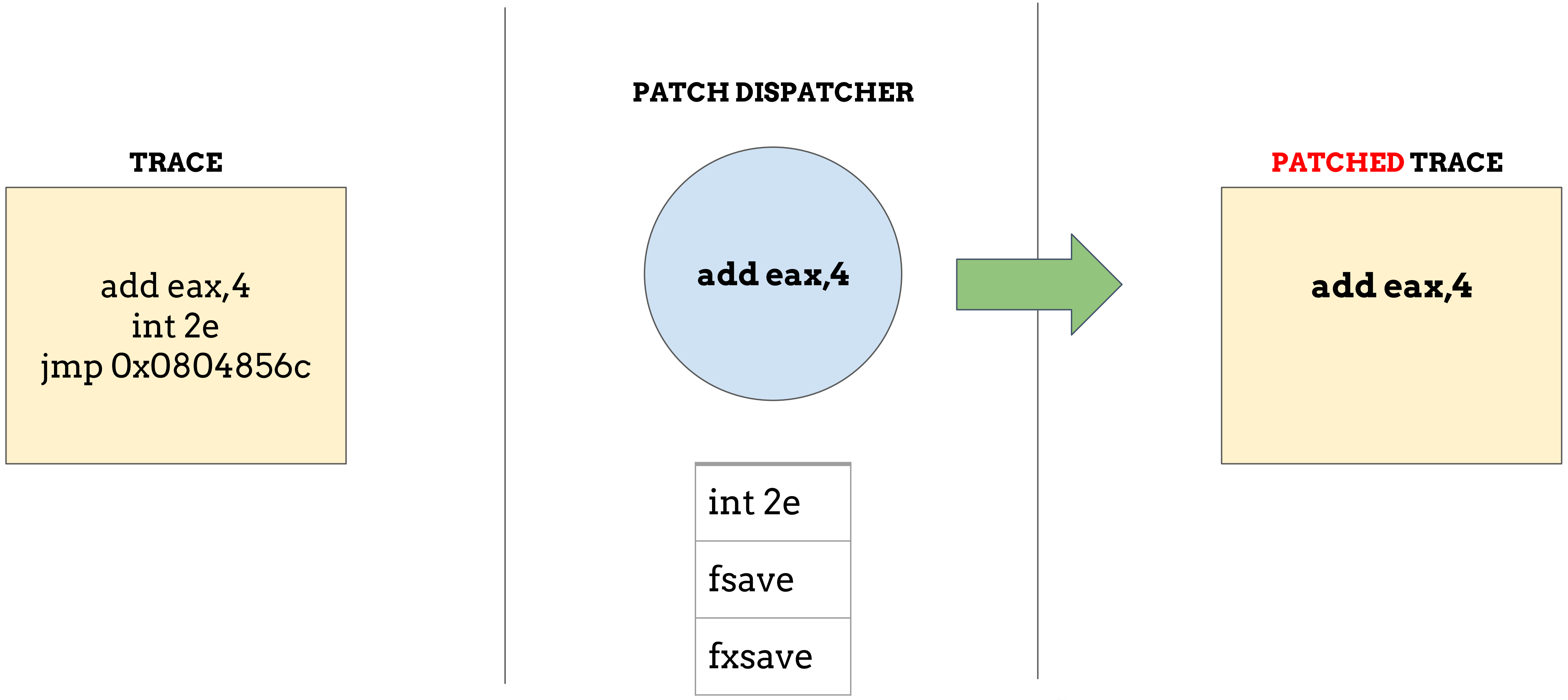
add eax,4
int 2e
jmp 0x0804856c

PATCH DISPATCHER



PATCHED TRACE

CCA - IP Detection



TRACE

```
add eax,4
int 2e
jmp 0x0804856c
```

PATCH DISPATCHER

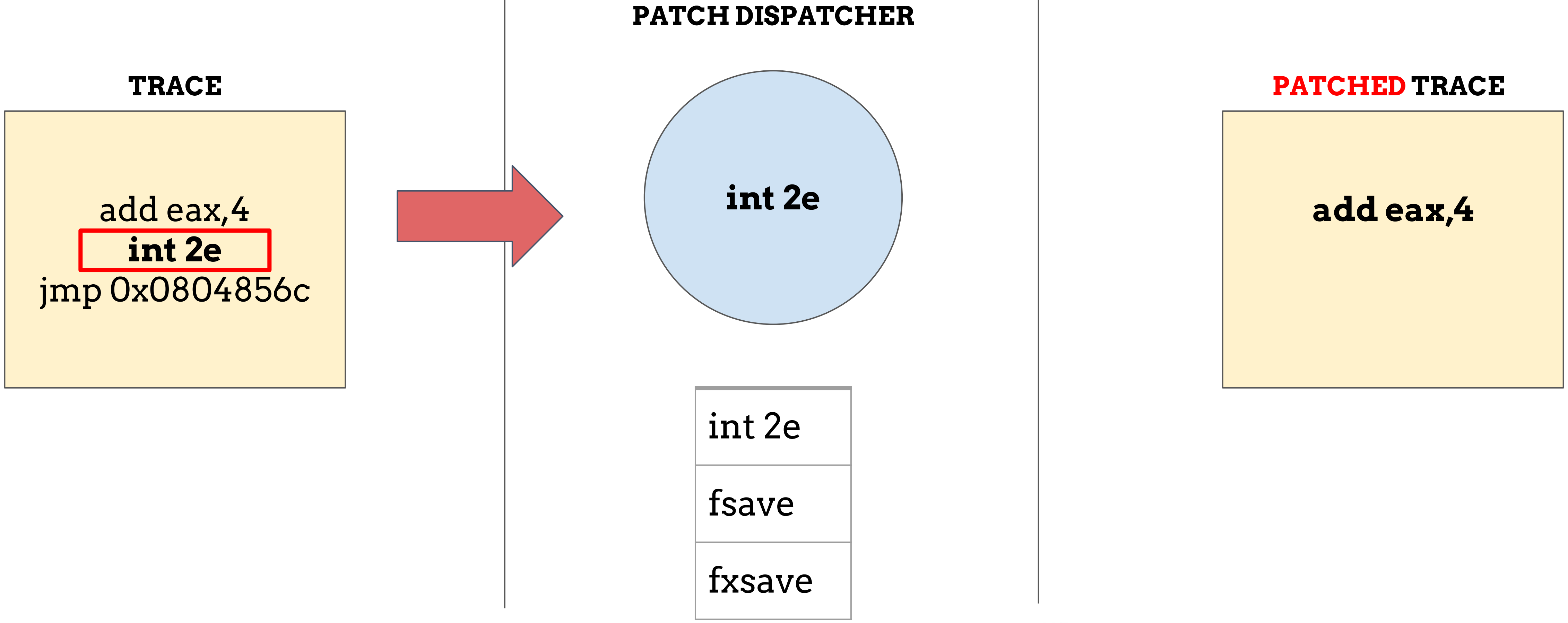
```
add eax,4
```

```
int 2e
fsave
fxsave
```

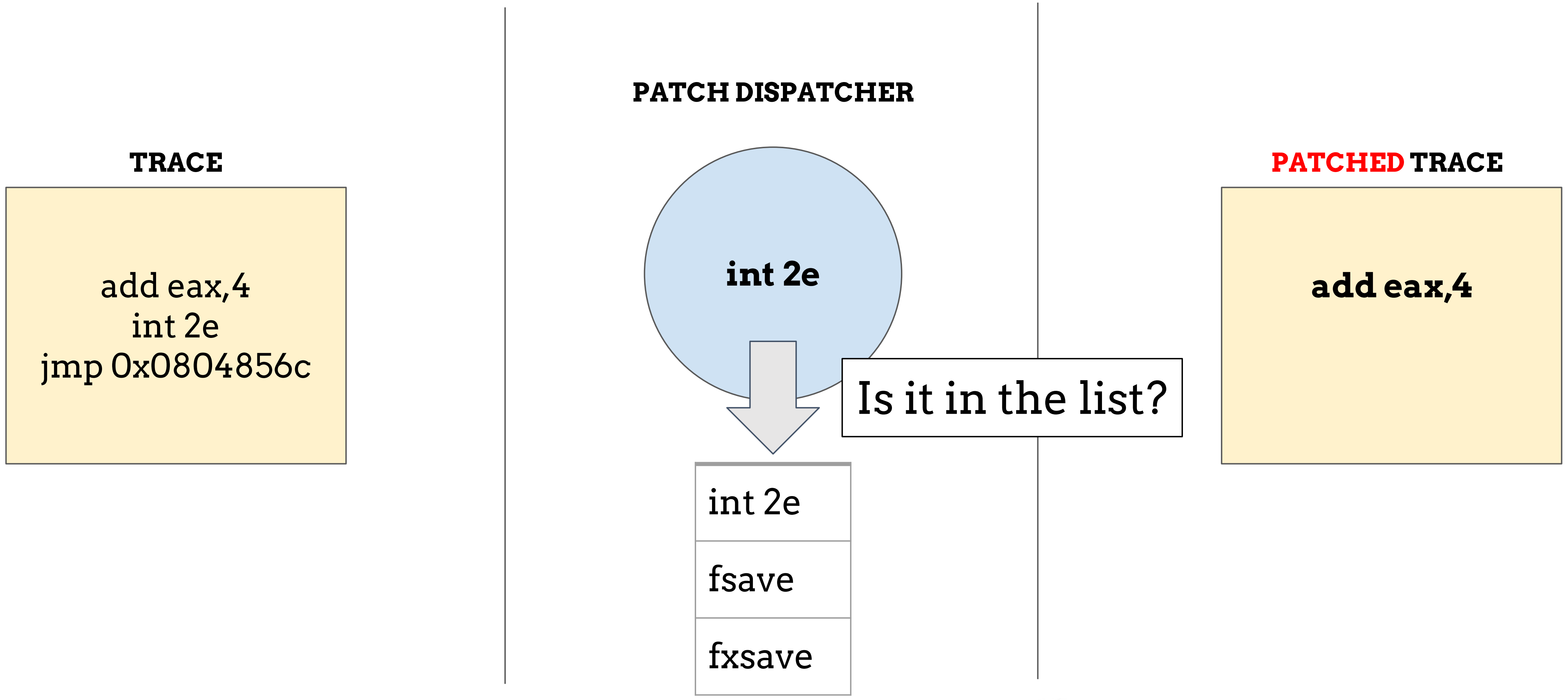
PATCHED TRACE

```
add eax,4
```

CCA - IP Detection



CCA - IP Detection



TRACE

```
add eax,4
int 2e
jmp 0x0804856c
```

PATCH DISPATCHER

```
int 2e
```

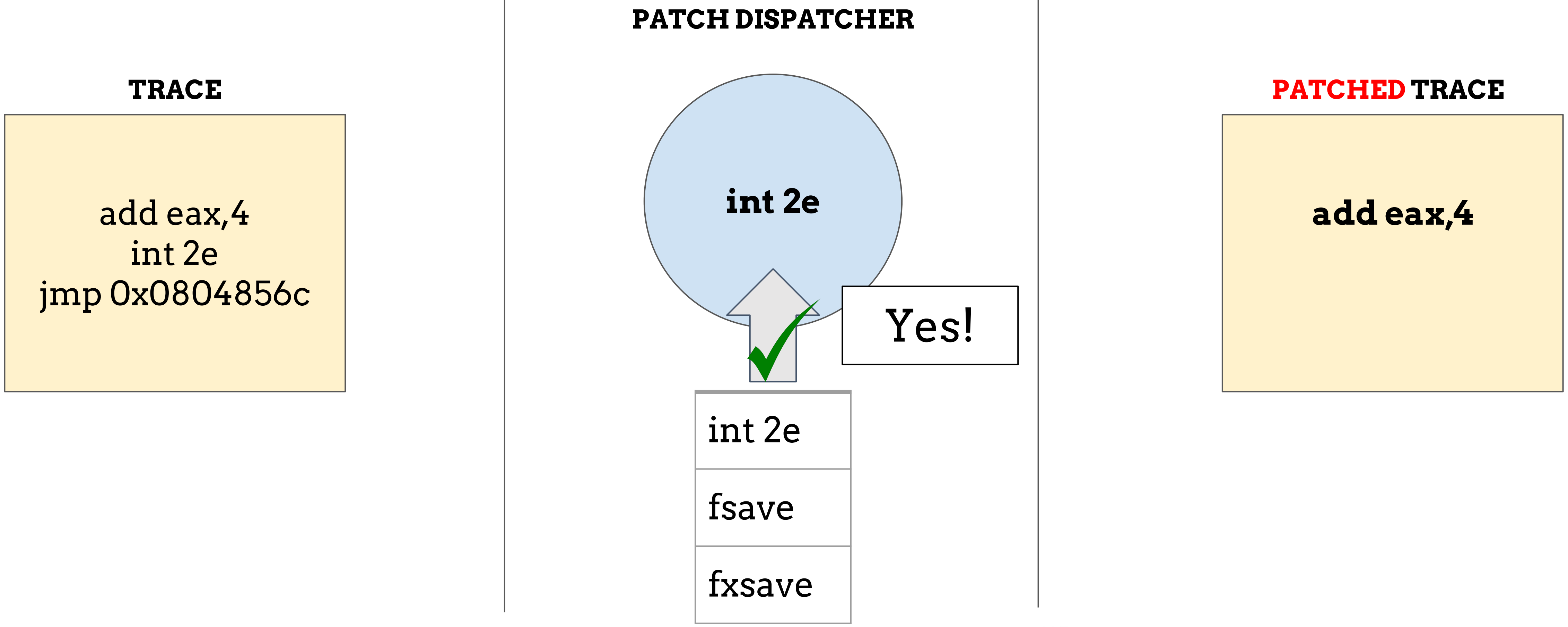
int 2e
fsave
fxsave

Is it in the list?

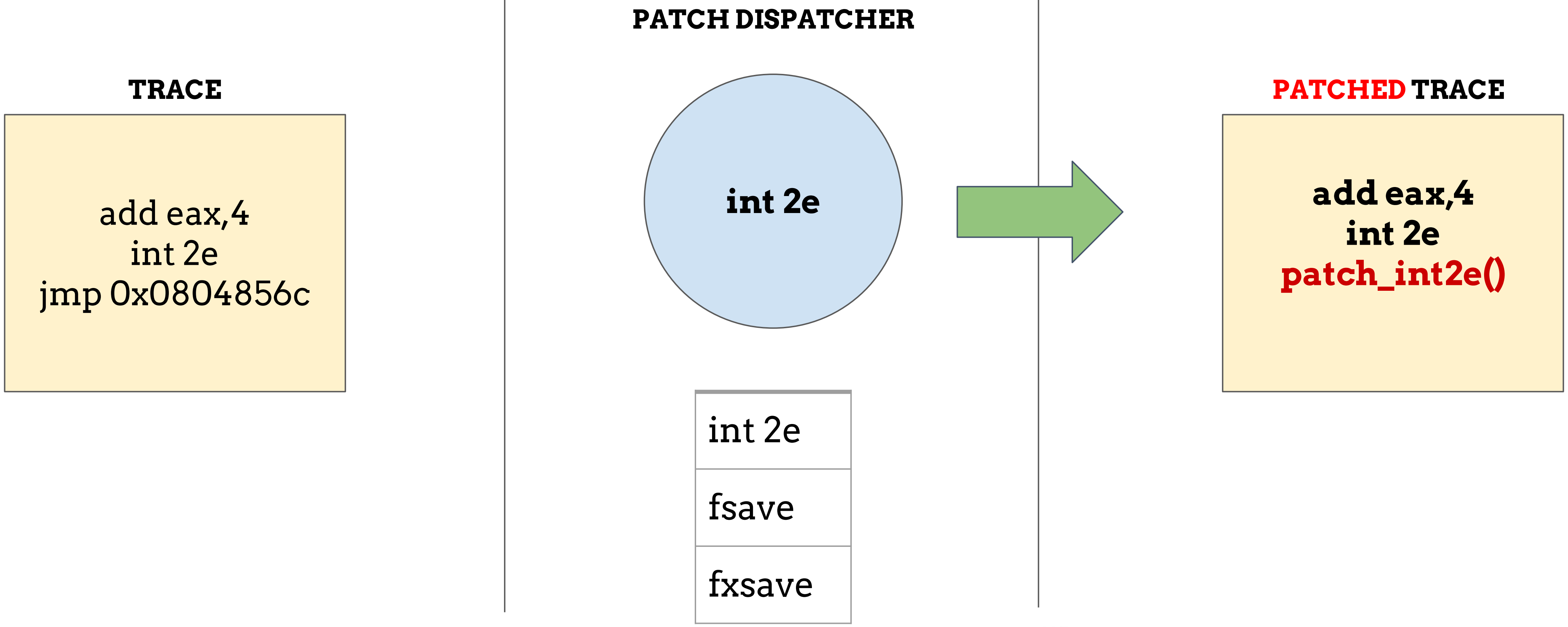
PATCHED TRACE

```
add eax,4
```

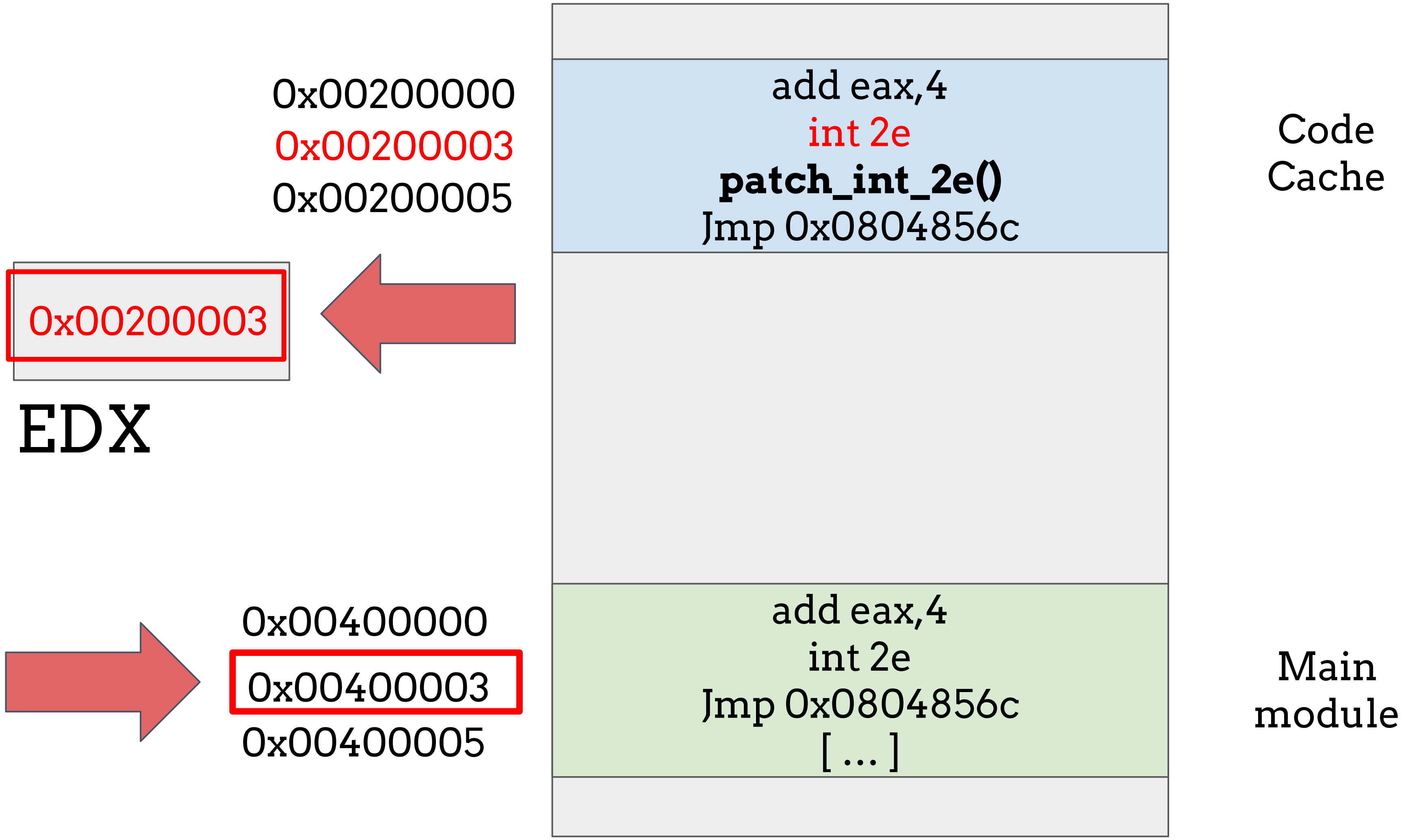
CCA - IP Detection



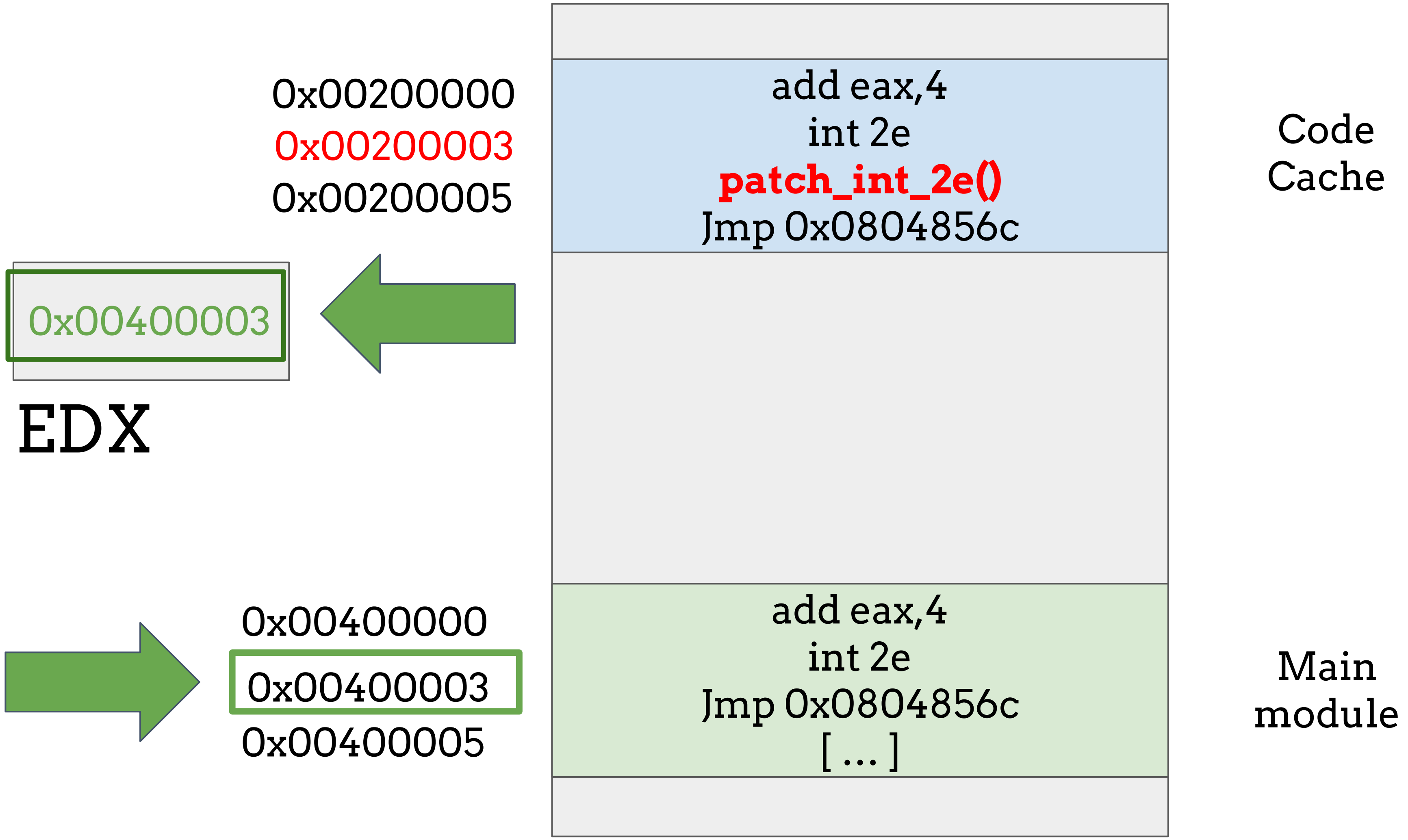
CCA - IP Detection



CCA - IP Detection RT



CCA - IP Detection RT

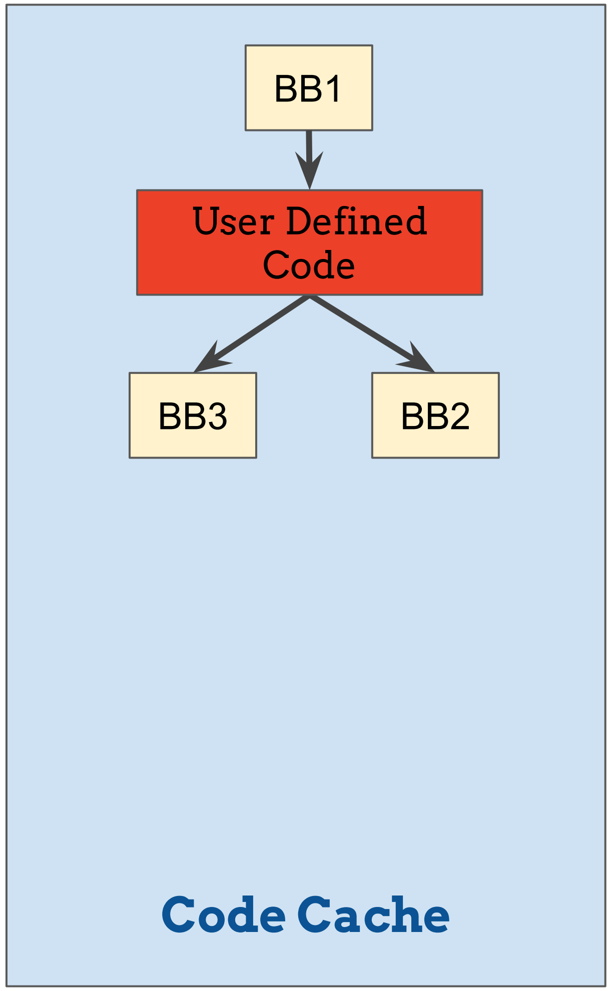


black hat EUROPE 2017

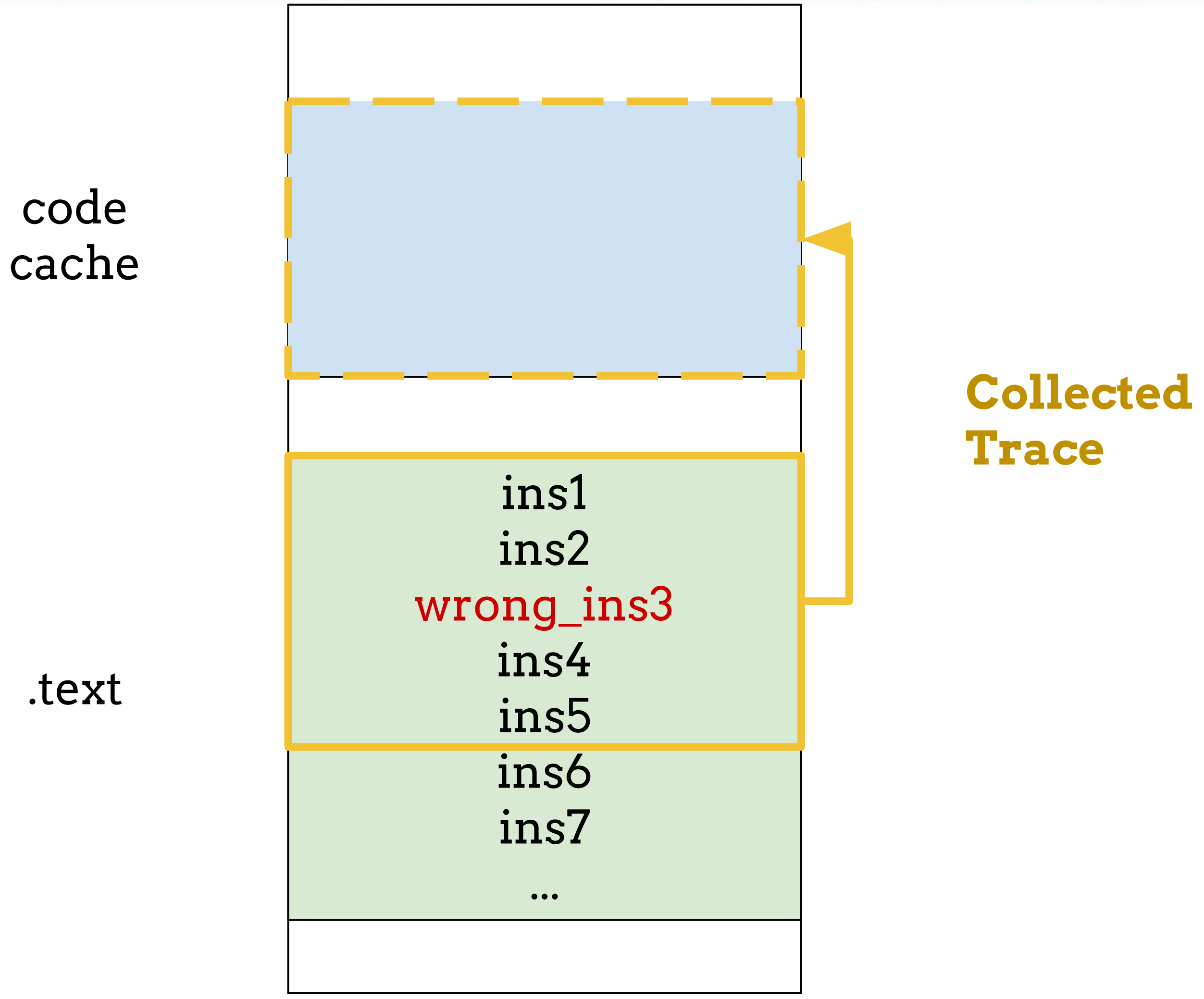
Code Cache Artifacts

All those artifacts caused by having a Code Cache

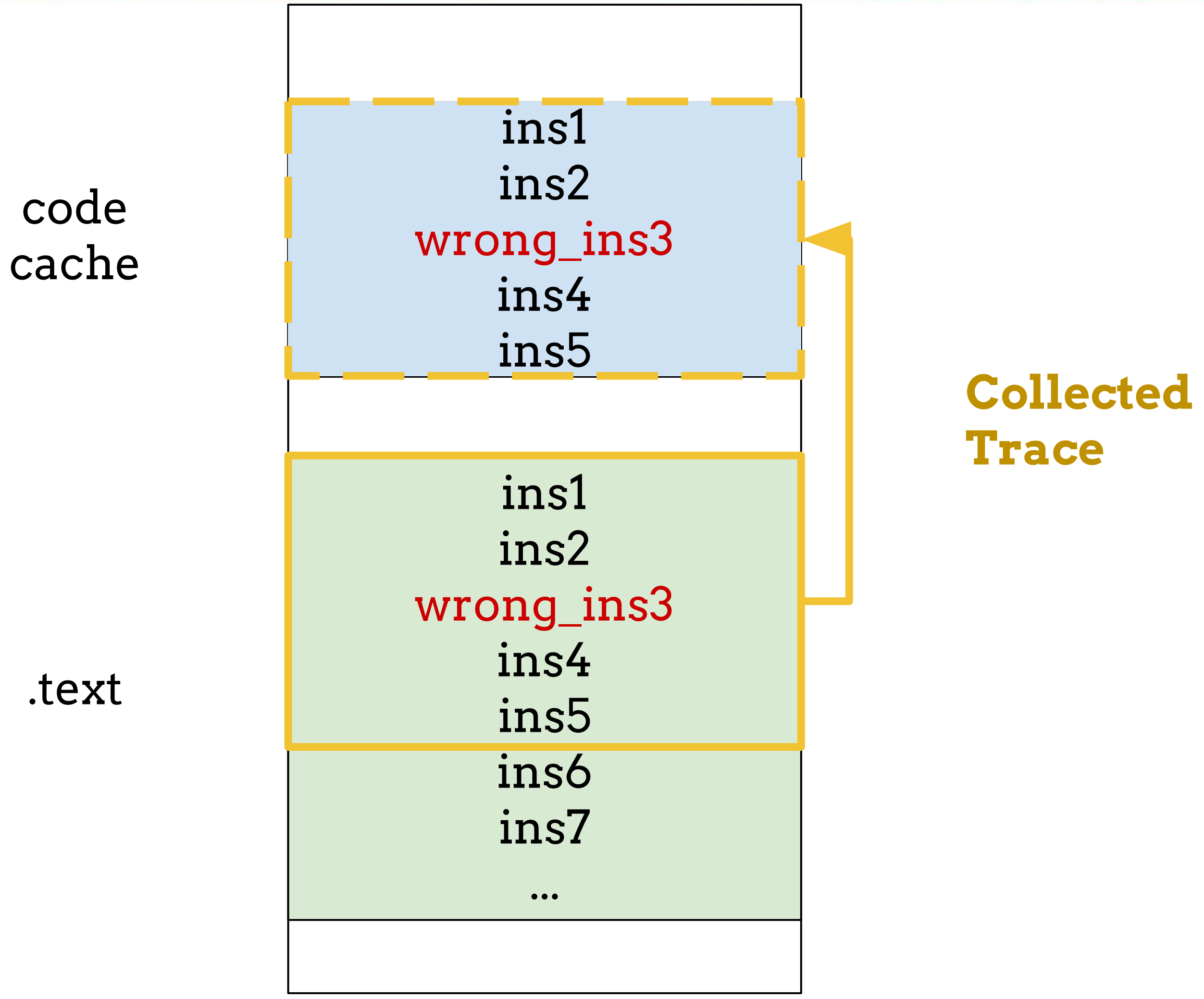
- **IP Detection**
- **Self-Modifying Code**



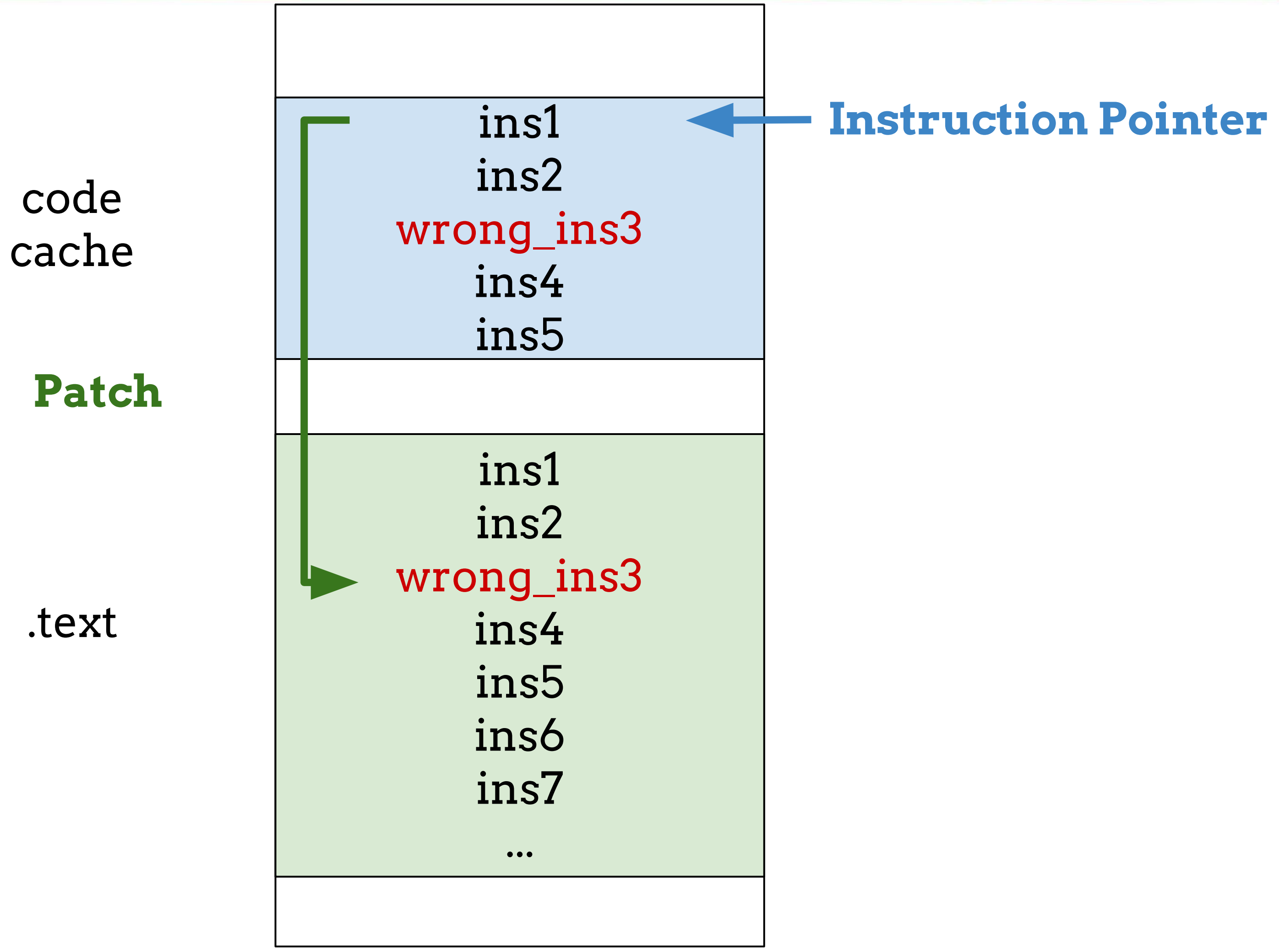
CCA - Self Modifying Code



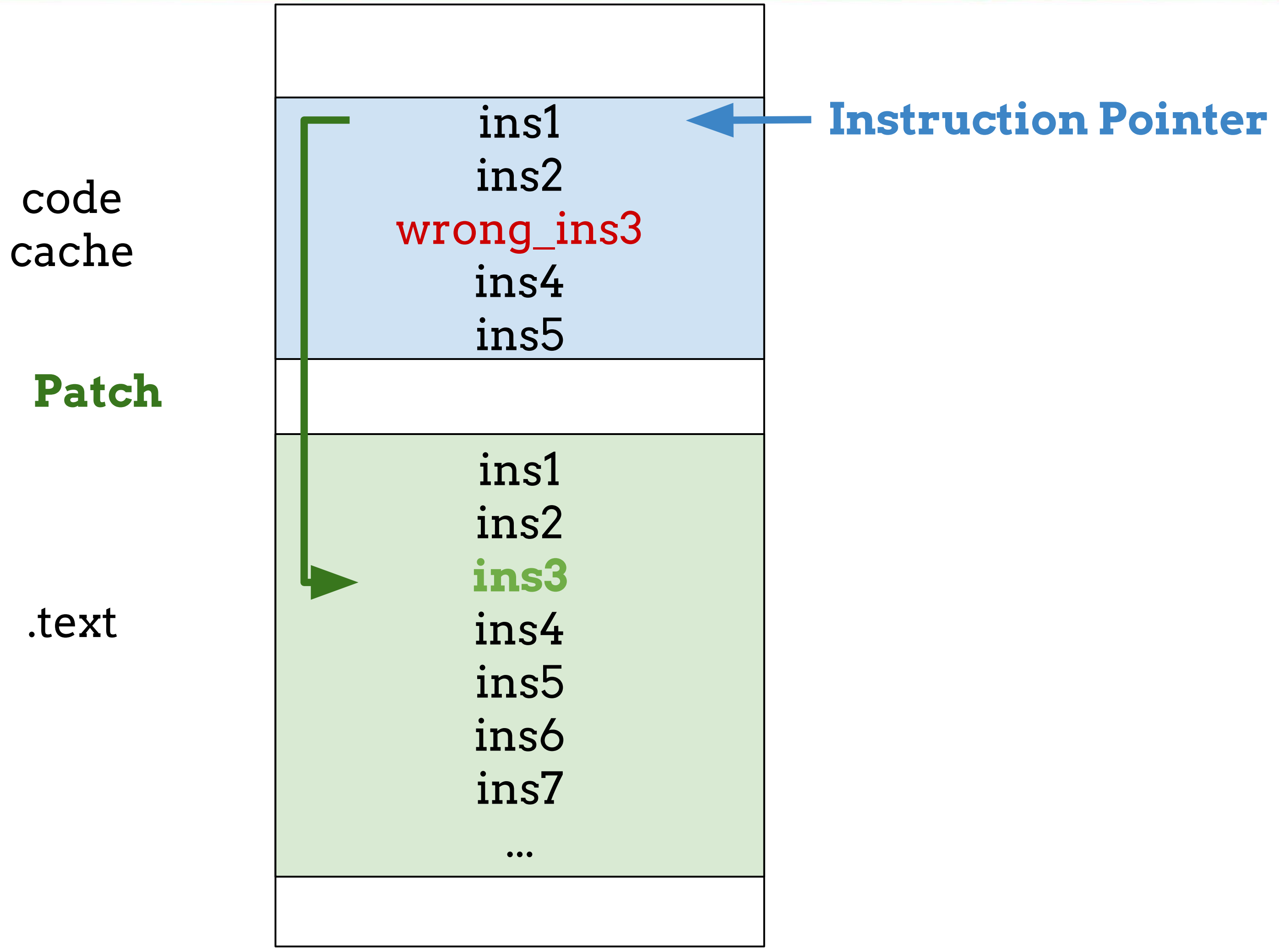
CCA - Self Modifying Code



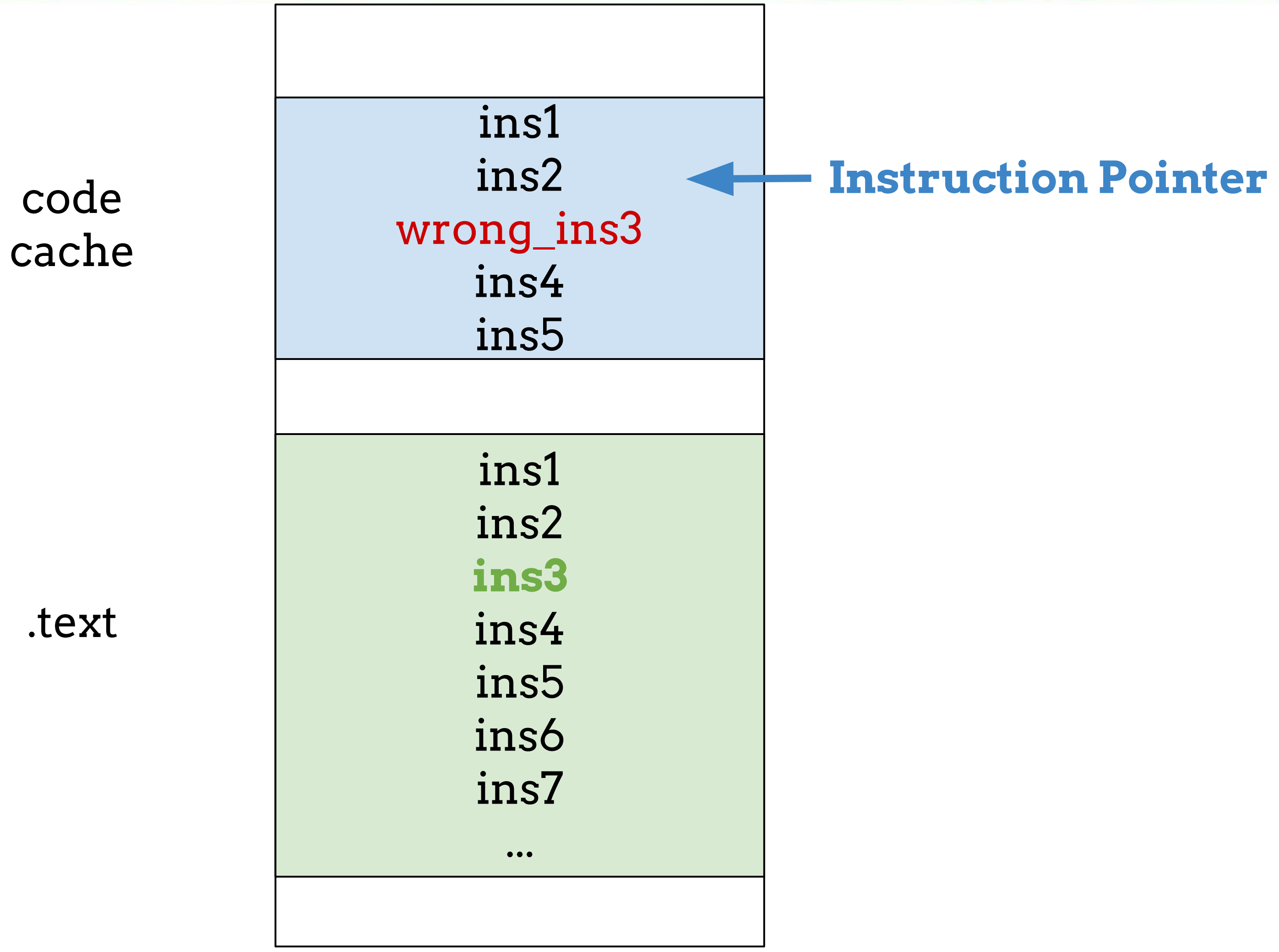
CCA - Self Modifying Code



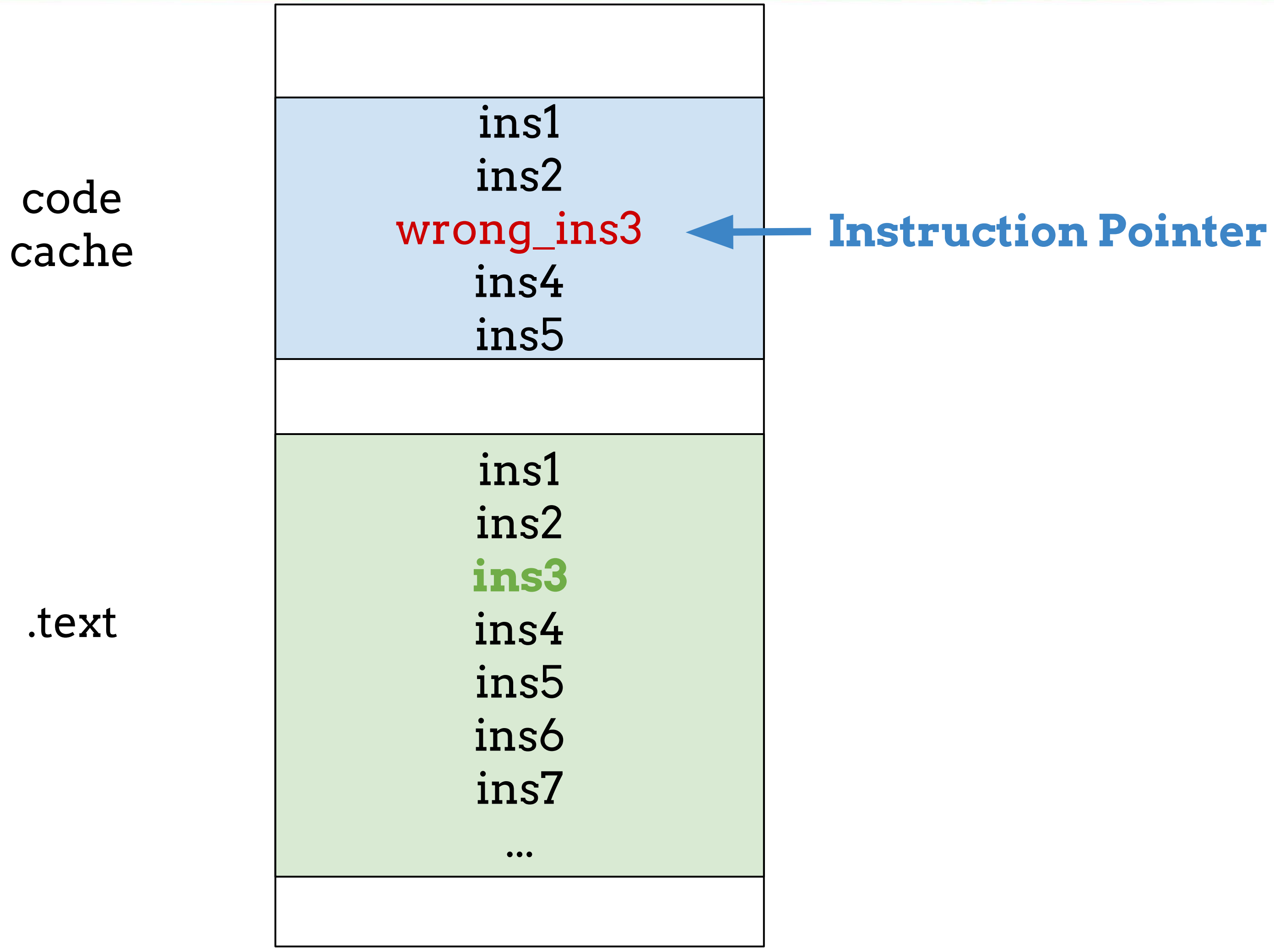
CCA - Self Modifying Code



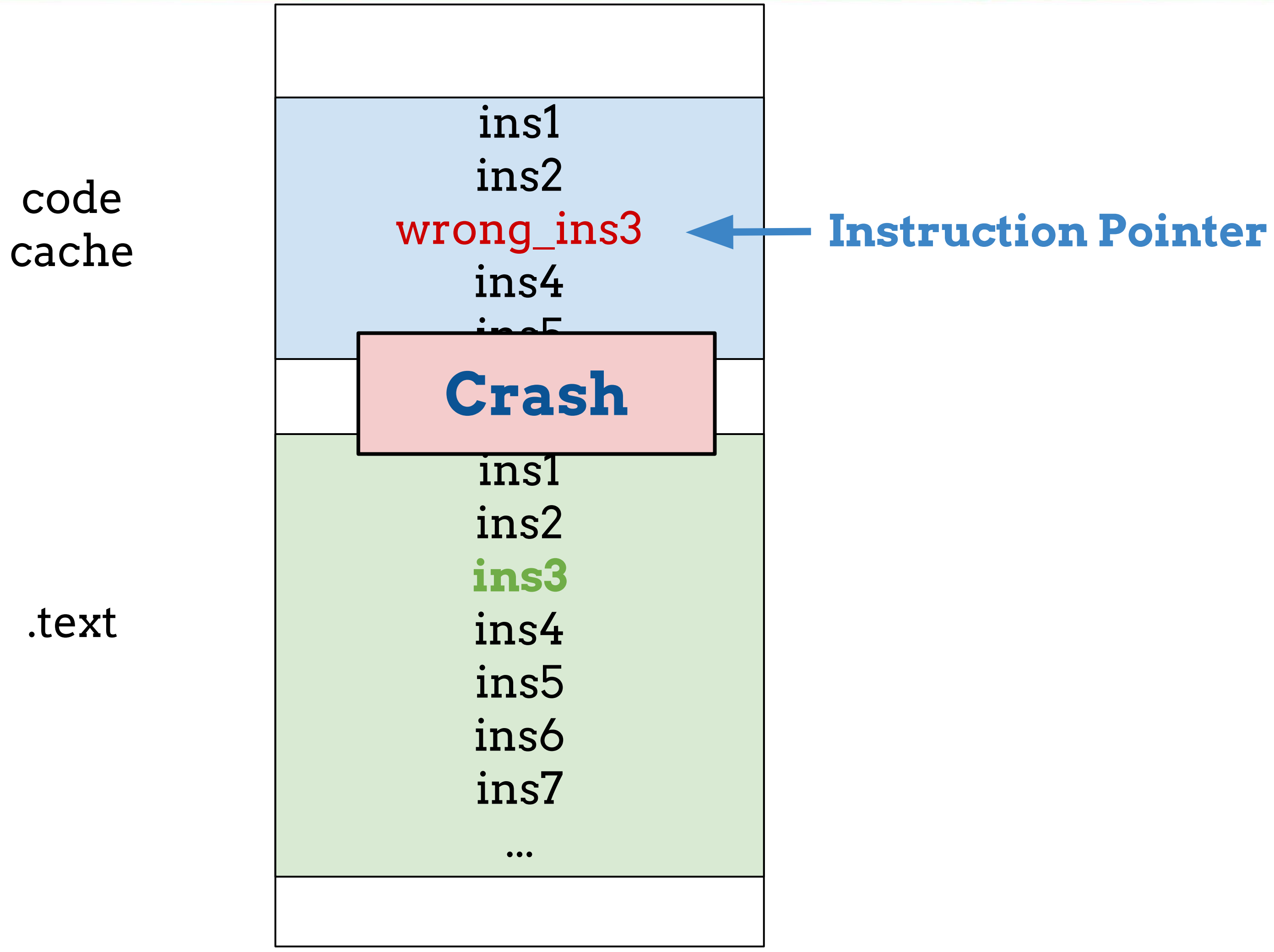
CCA - Self Modifying Code



CCA - Self Modifying Code



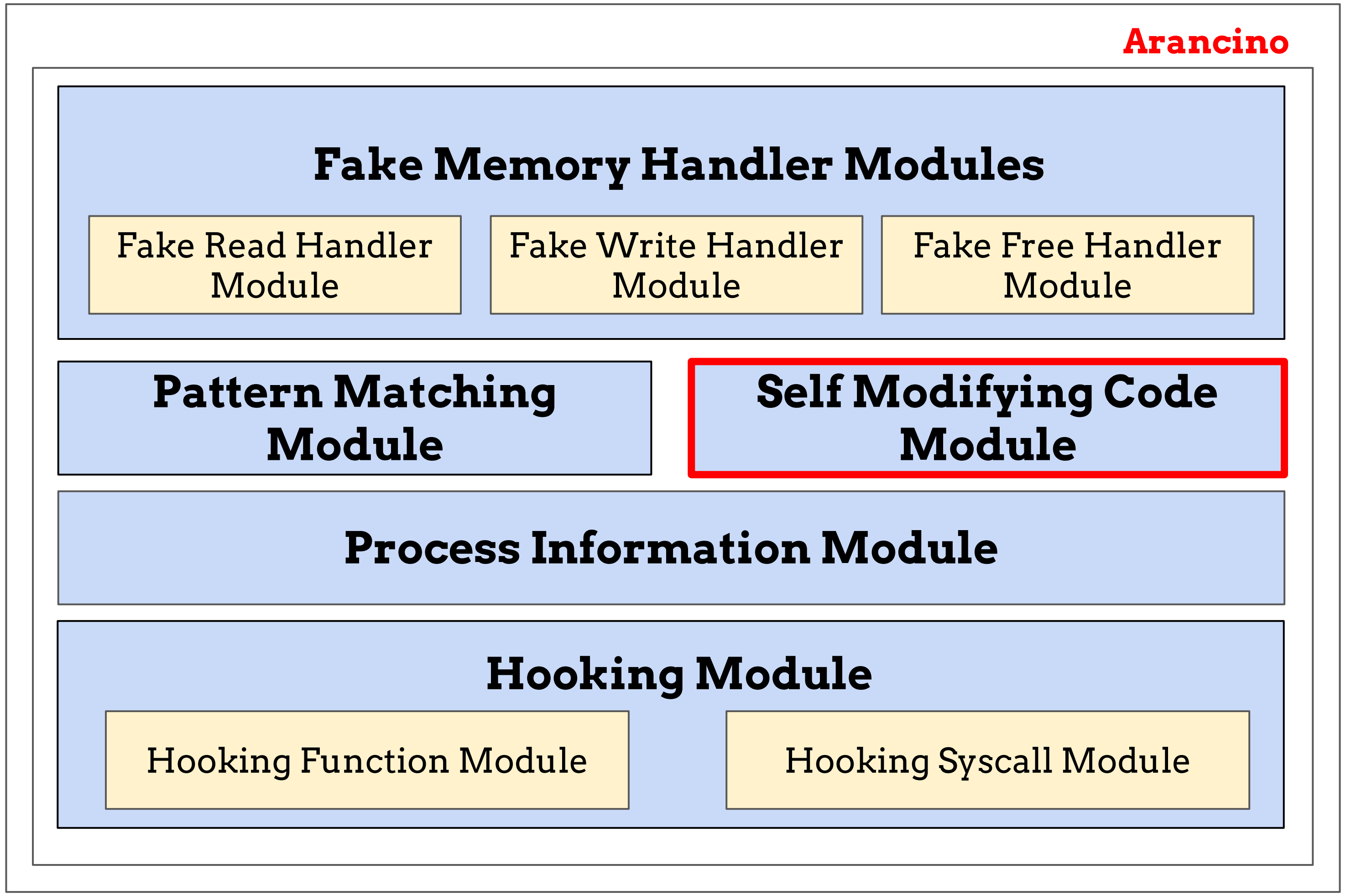
CCA - Self Modifying Code



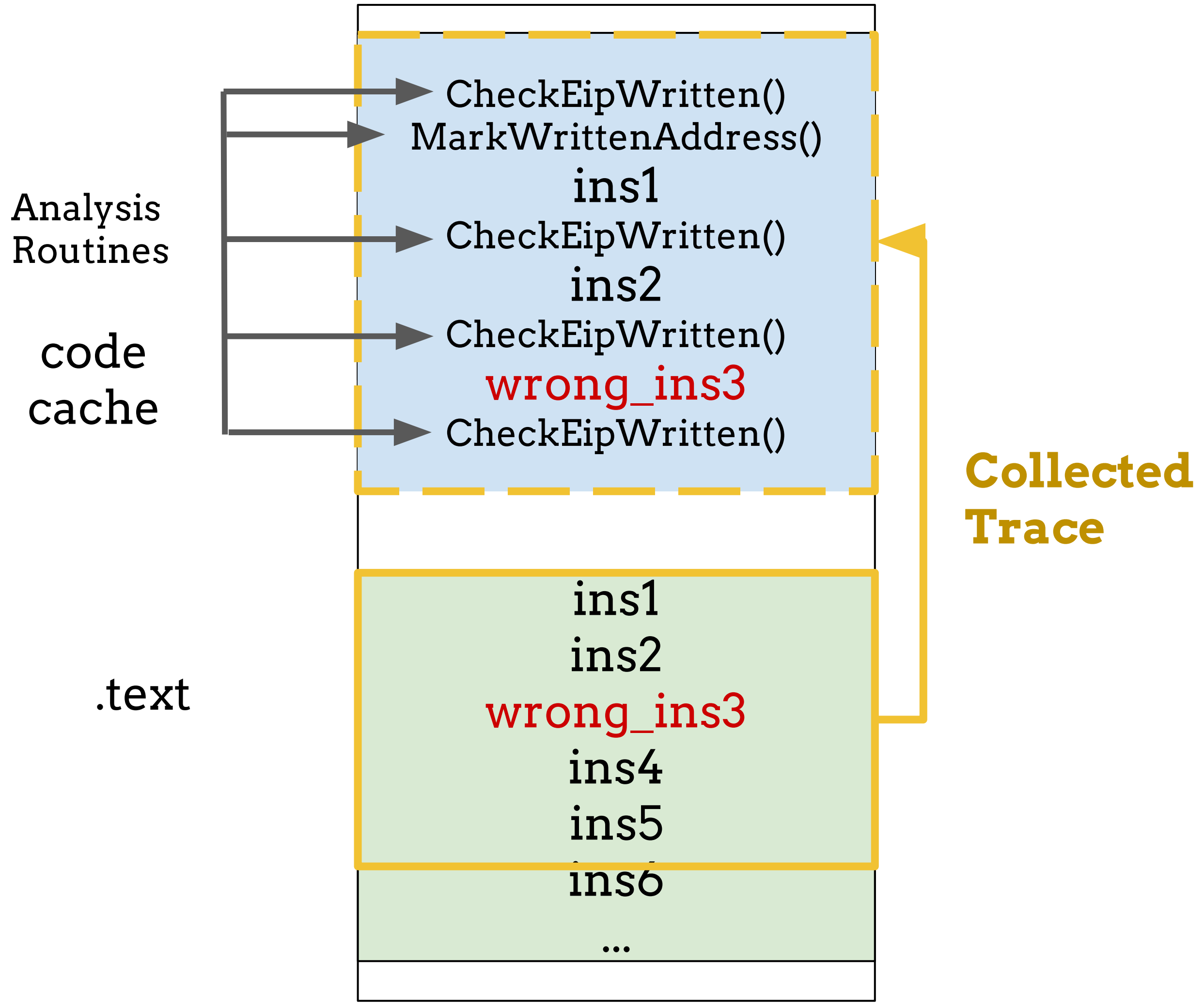
Arancino - Self Modifying Code Module

- **MarkWrittenAddress**: store which address has been overwritten

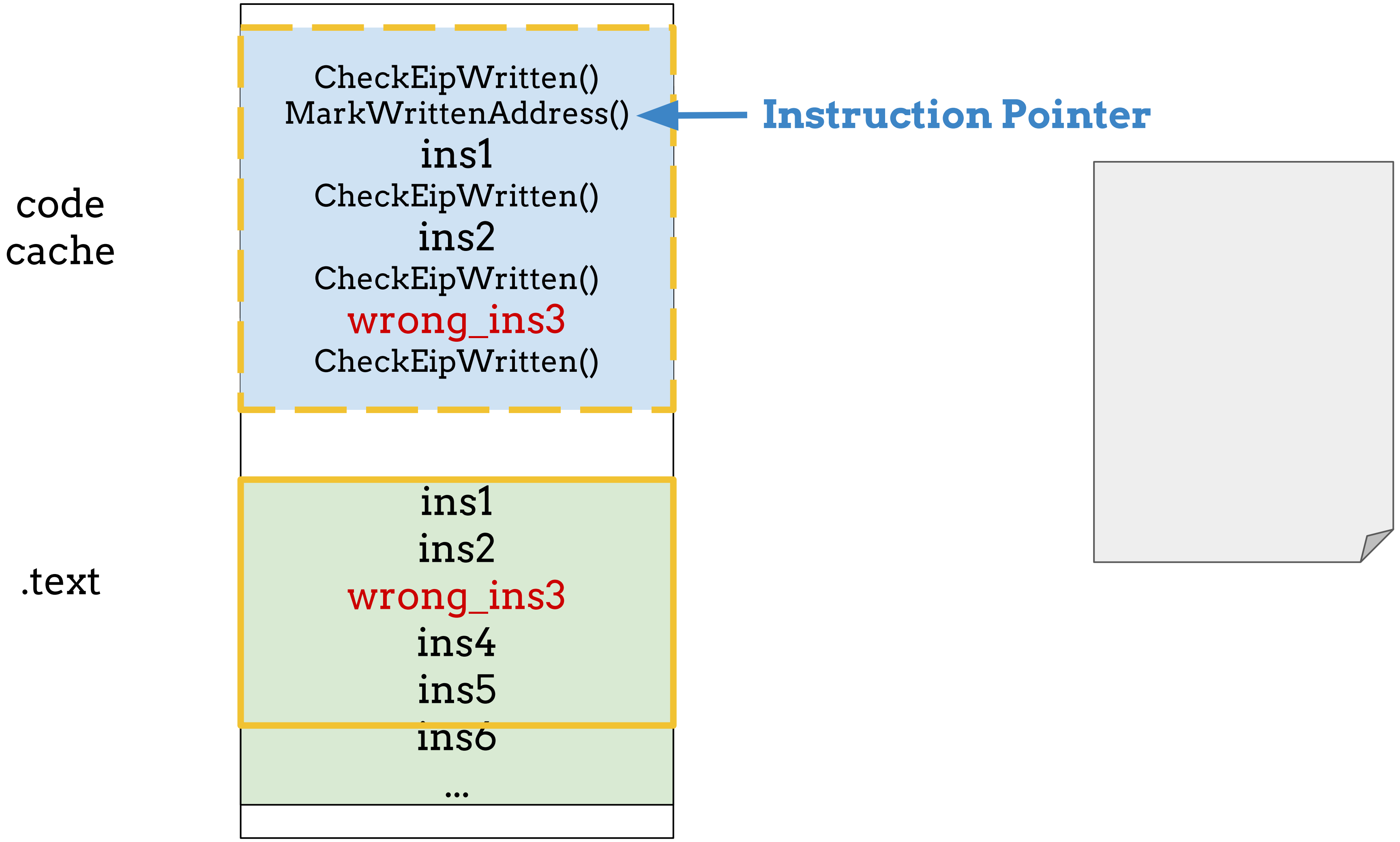
- **CheckEIPWritten**: check if next instruction has been overwritten.



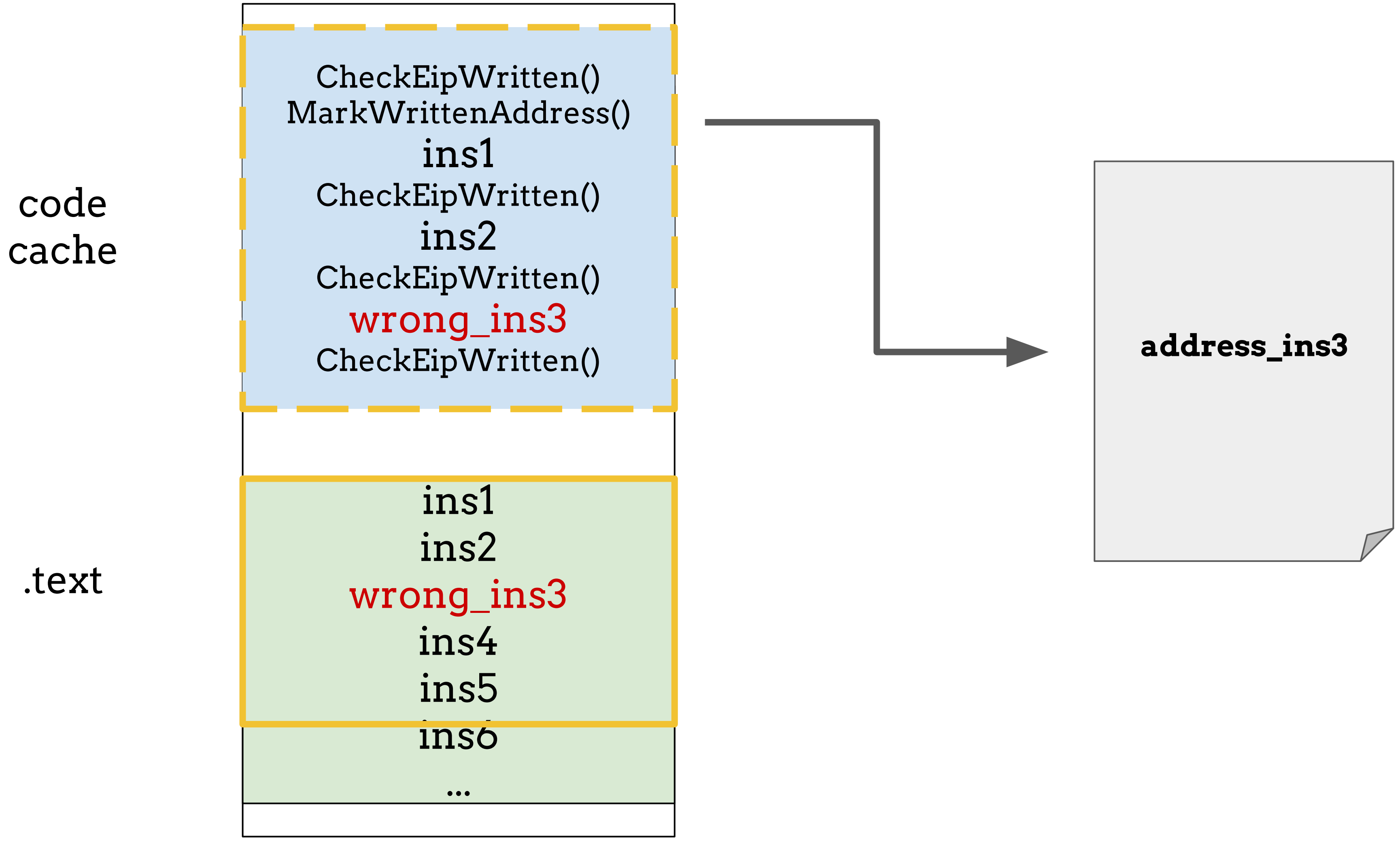
CCA - Self Modifying Code



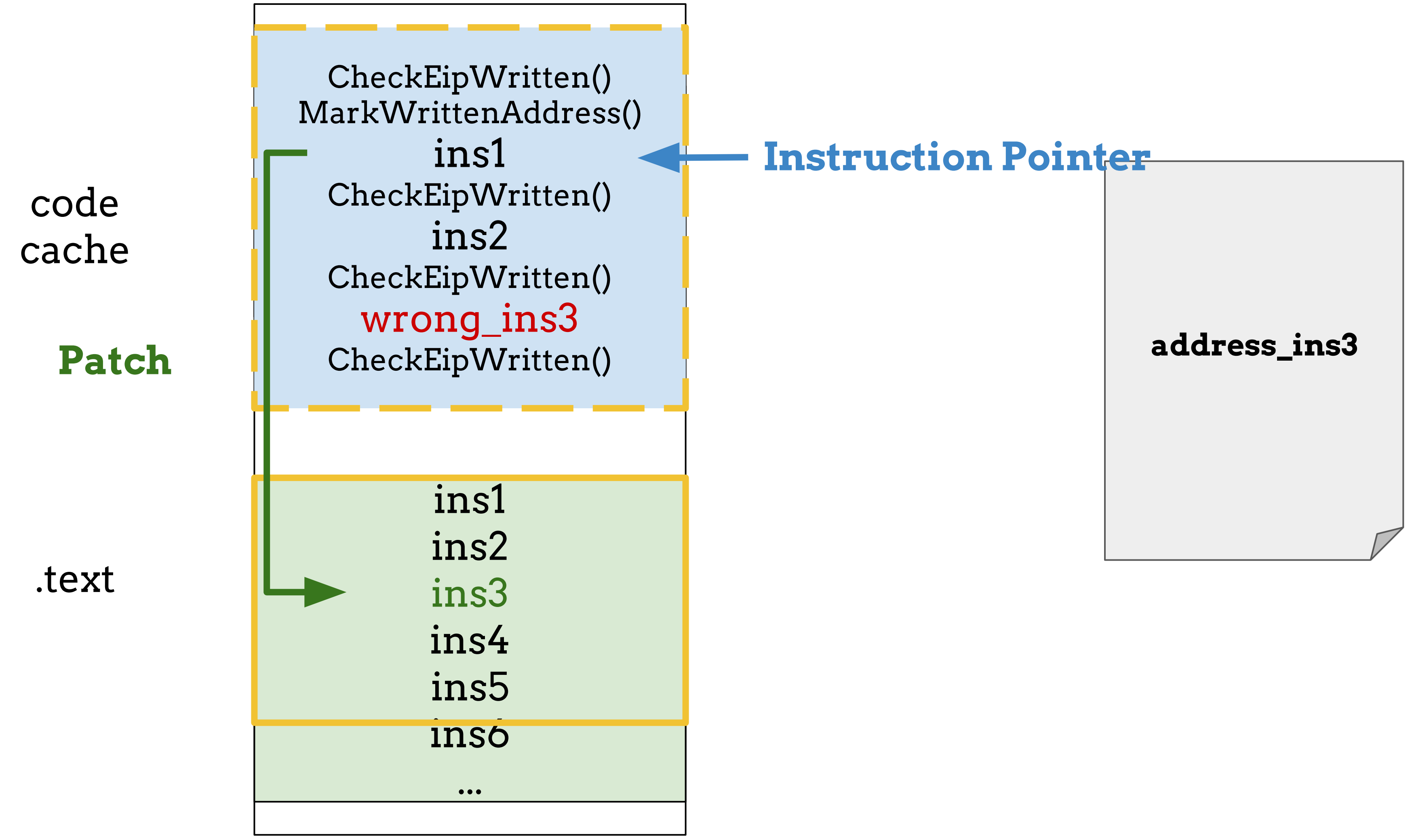
CCA - Self Modifying Code



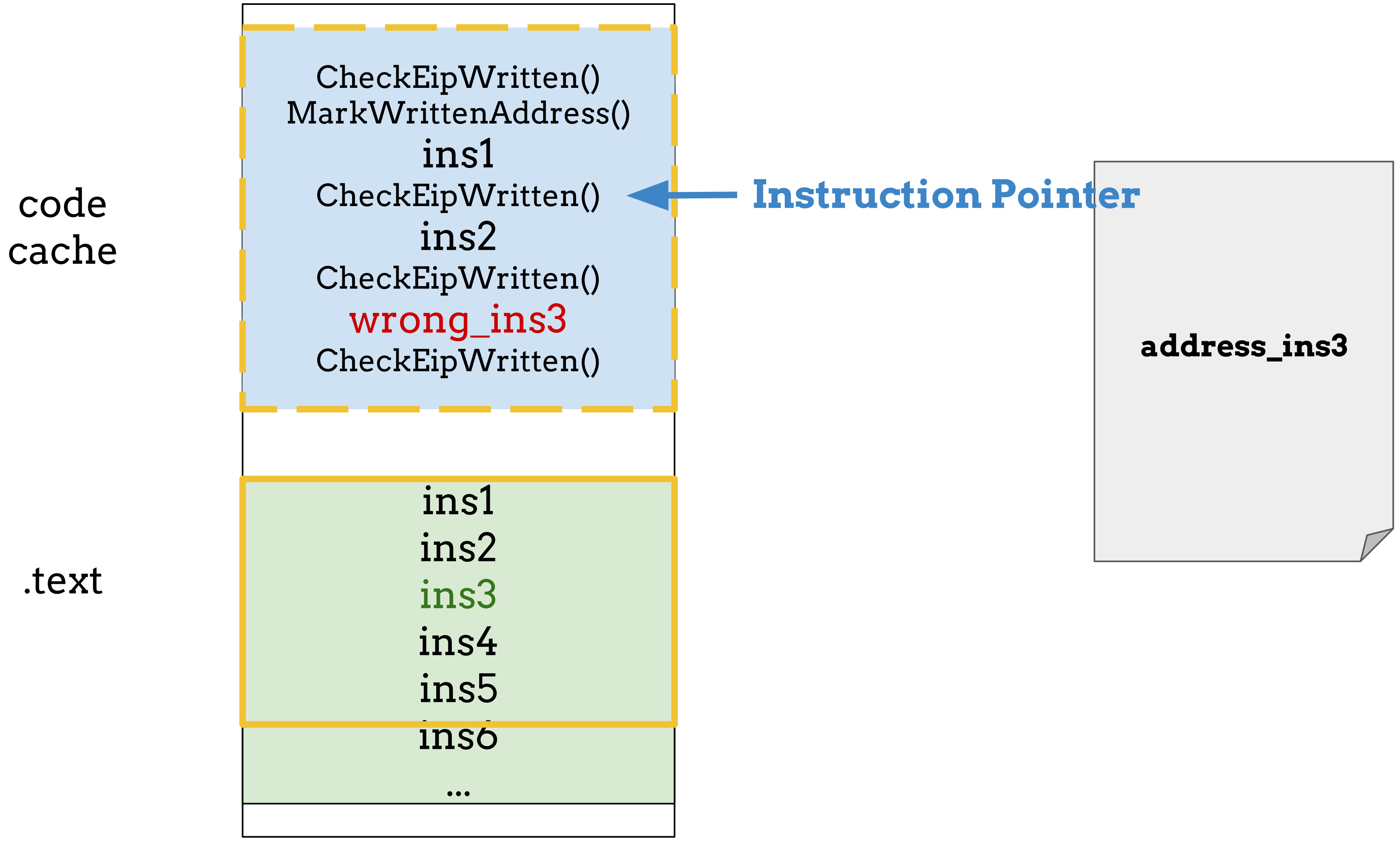
CCA - Self Modifying Code



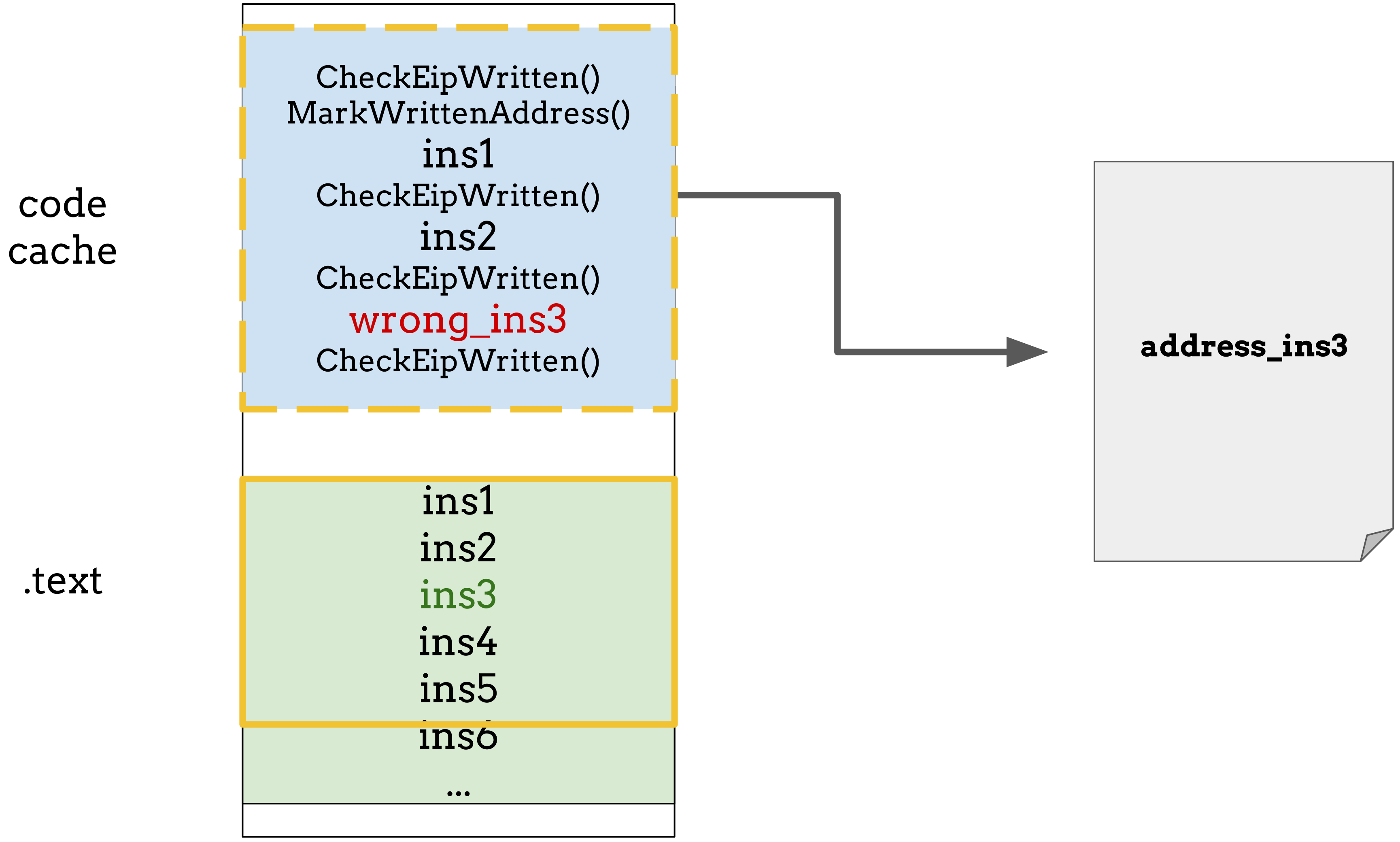
CCA - Self Modifying Code



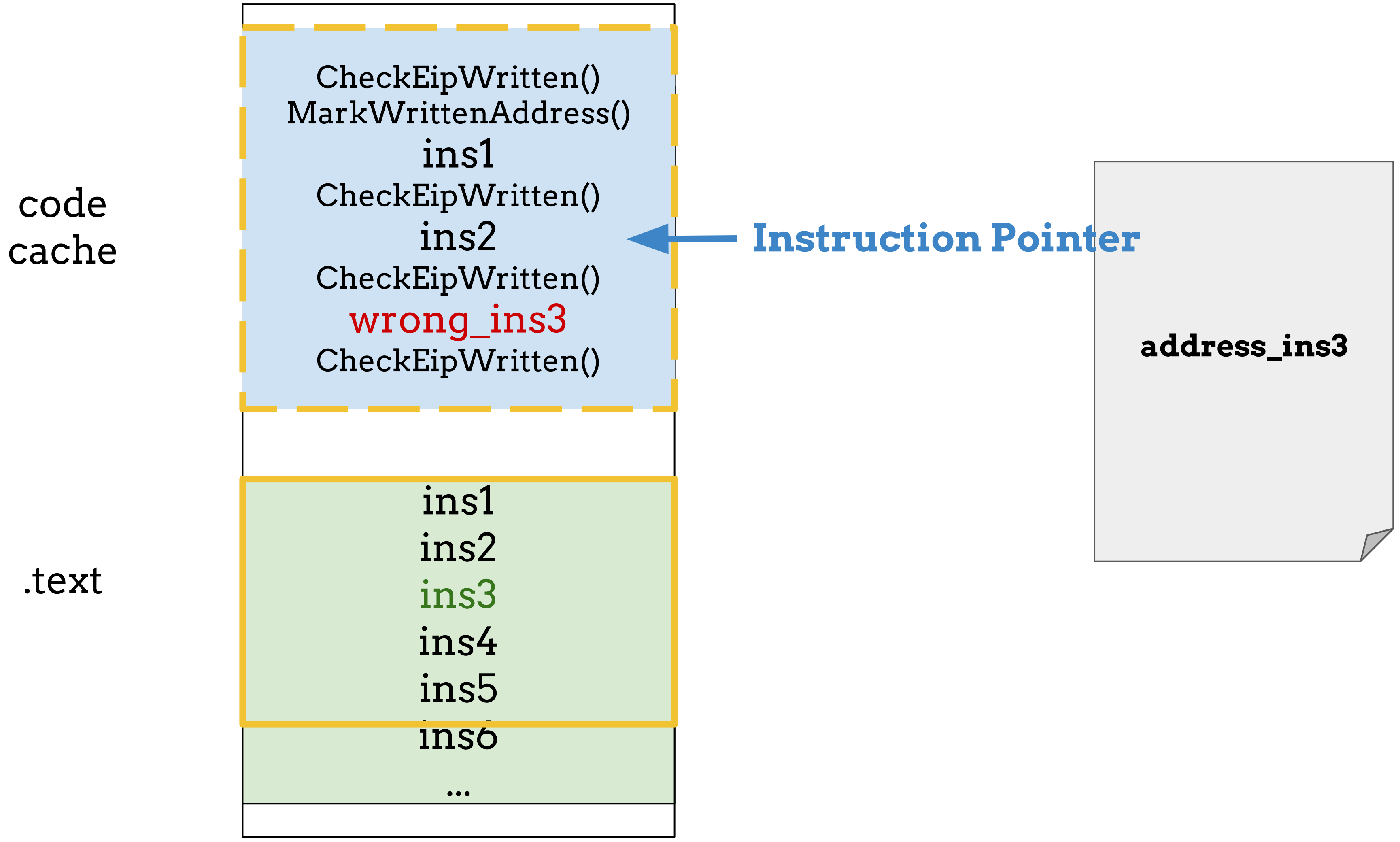
CCA - Self Modifying Code



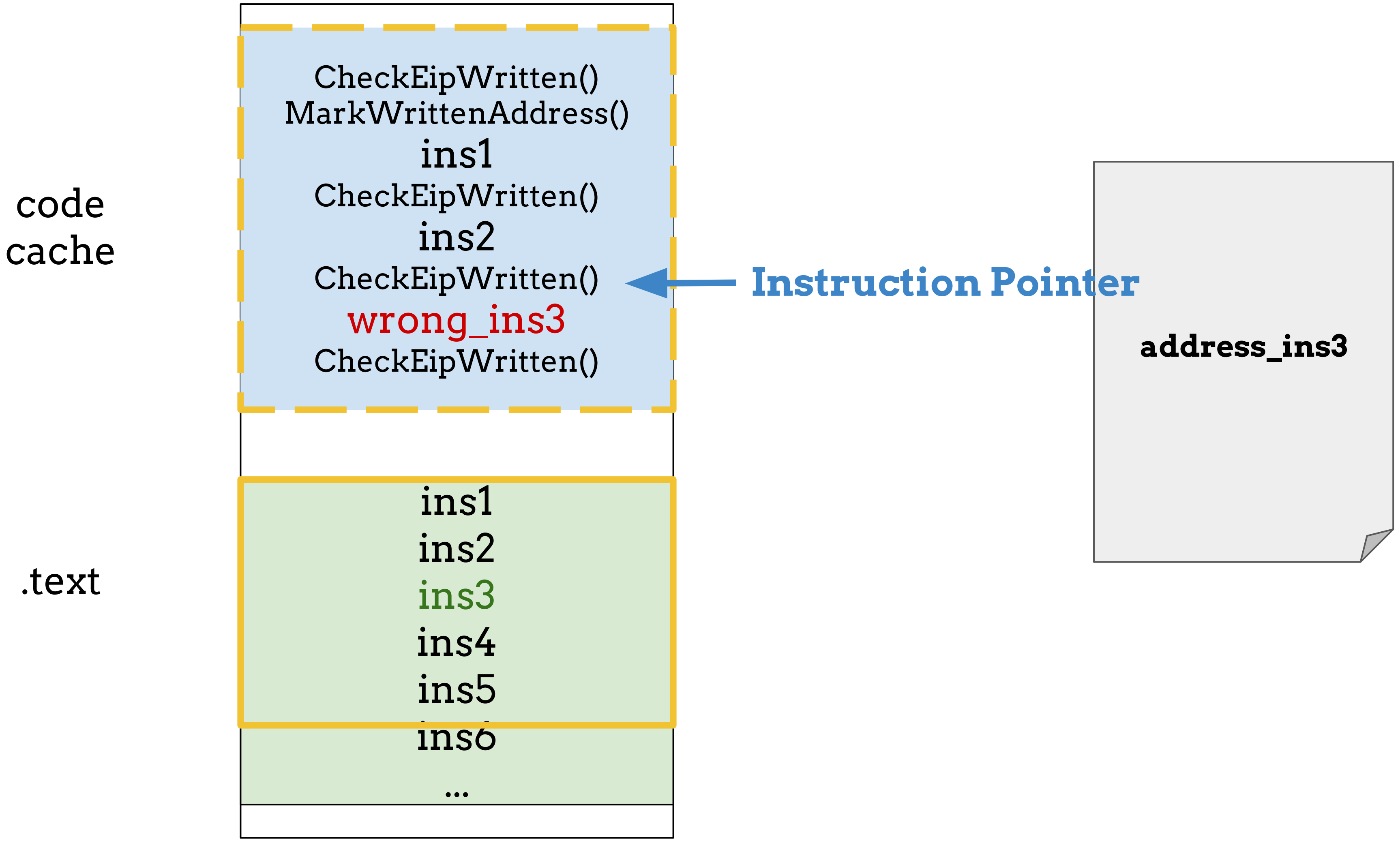
CCA - Self Modifying Code



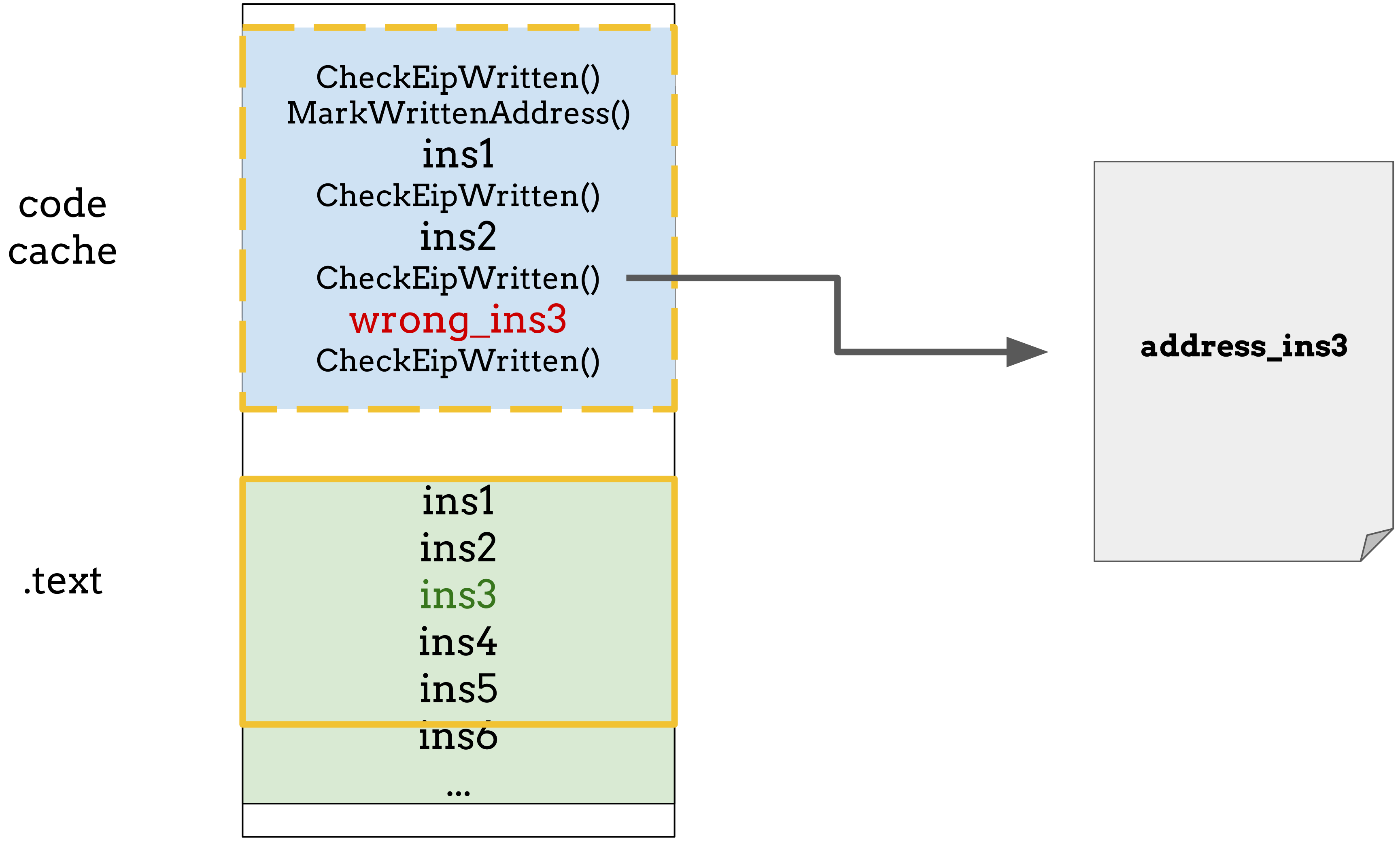
CCA - Self Modifying Code



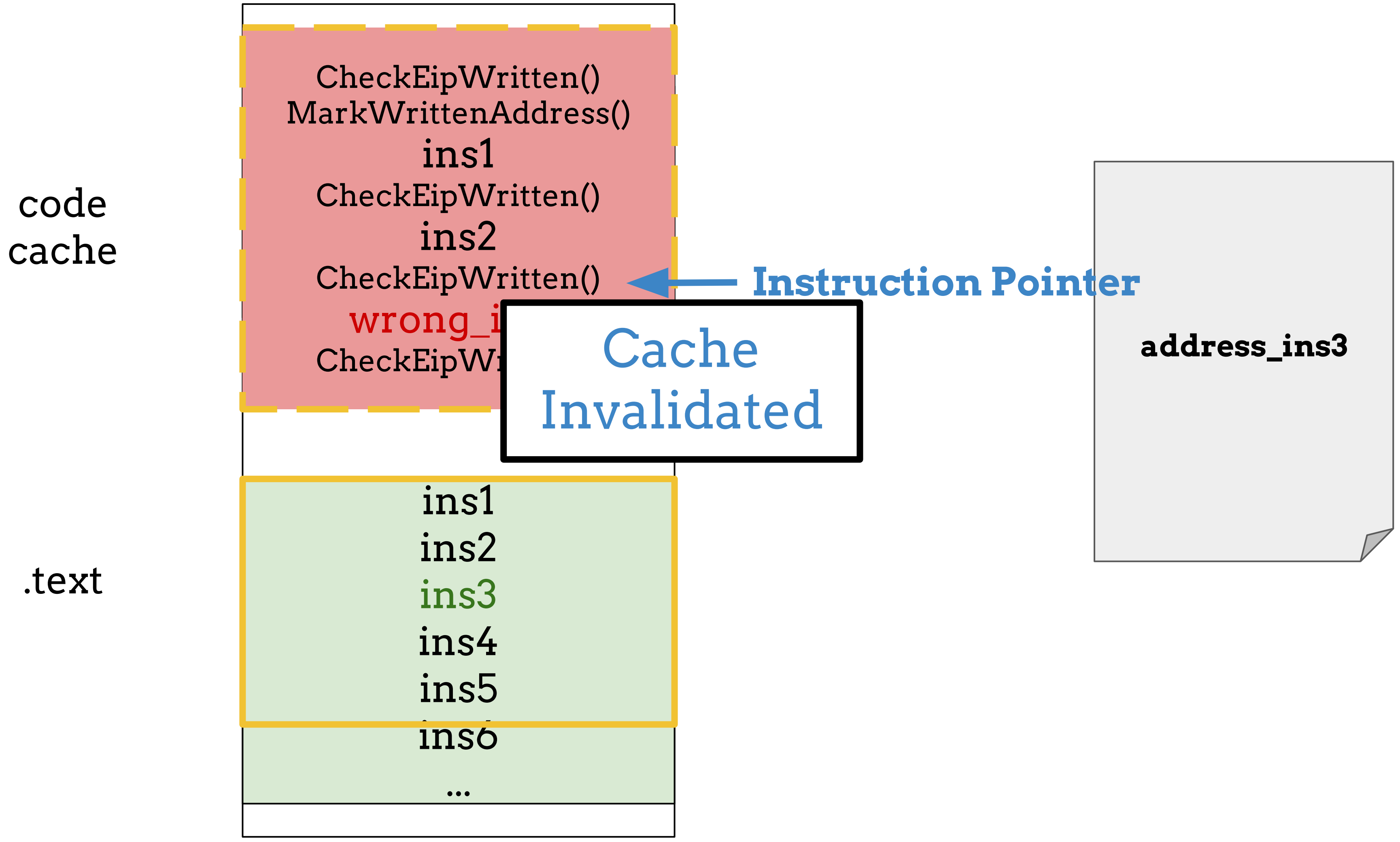
CCA - Self Modifying Code



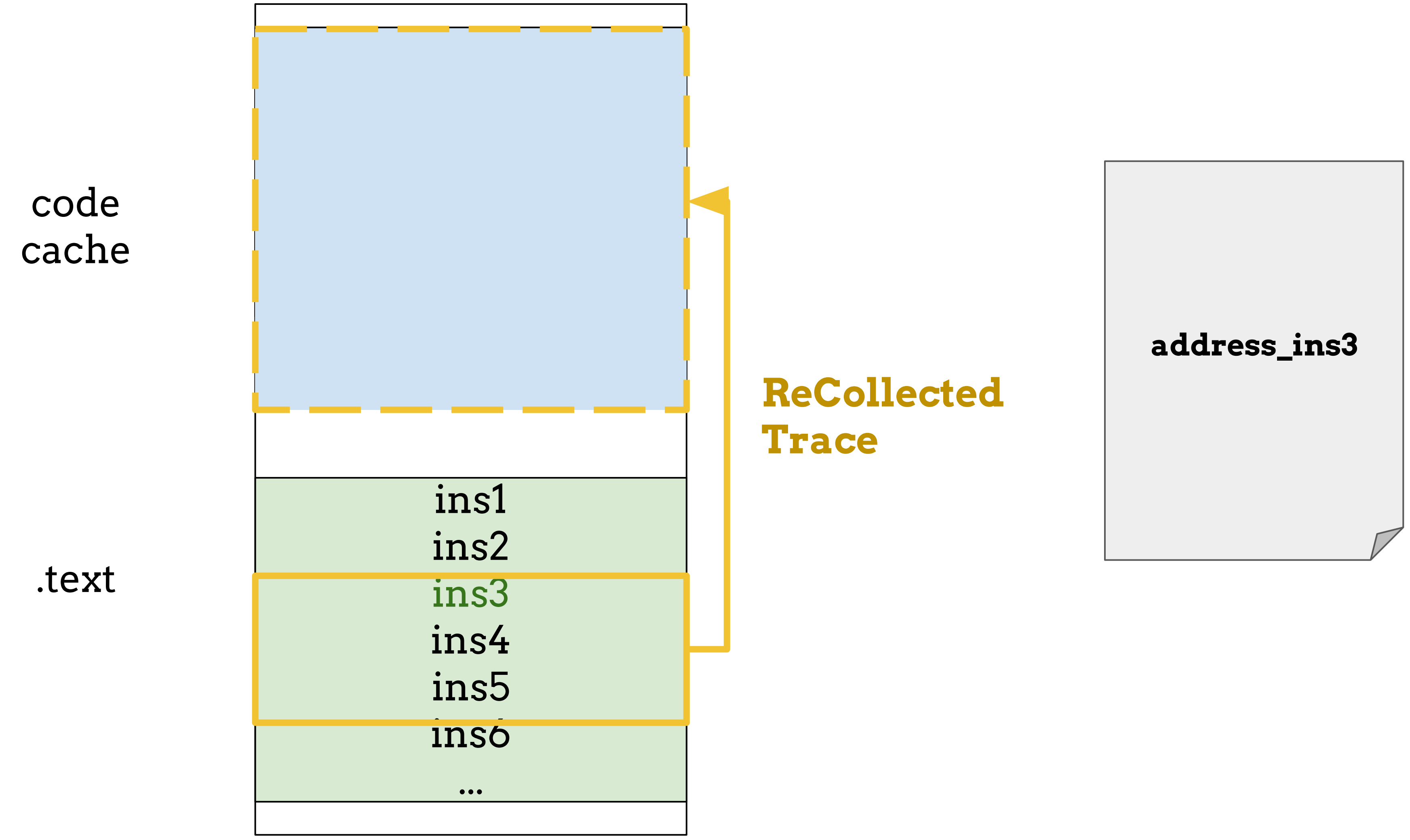
CCA - Self Modifying Code



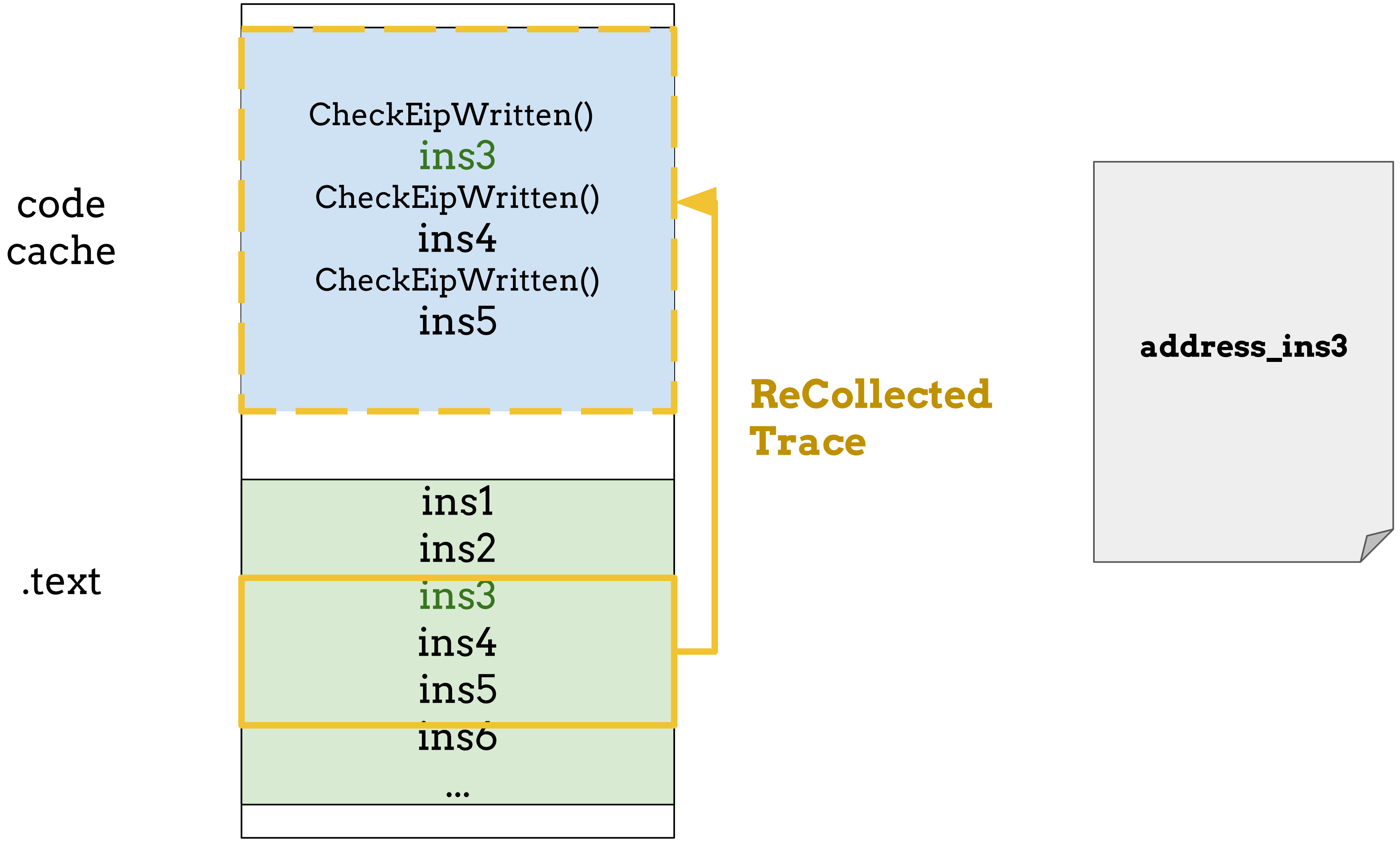
CCA - Self Modifying Code



CCA - Self Modifying Code



CCA - Self Modifying Code



Environment Artifacts

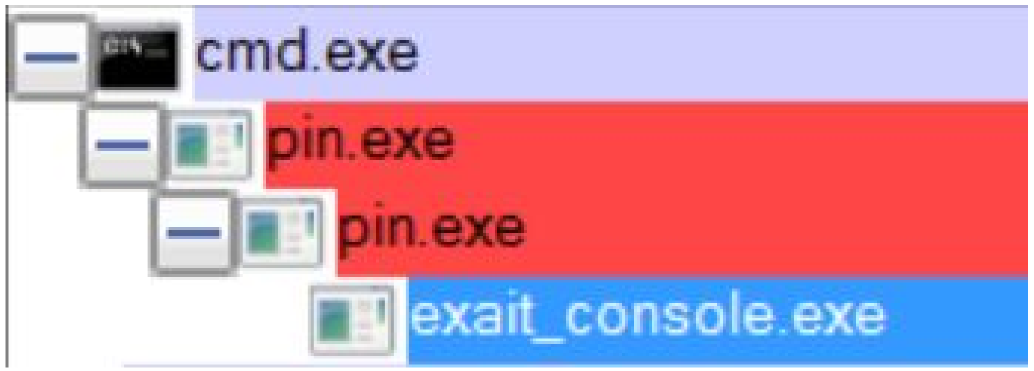
Environment Artifacts

- Parent Detection
- Memory Fingerprinting

EA - Parent Detection

Malware can check which is the process father.

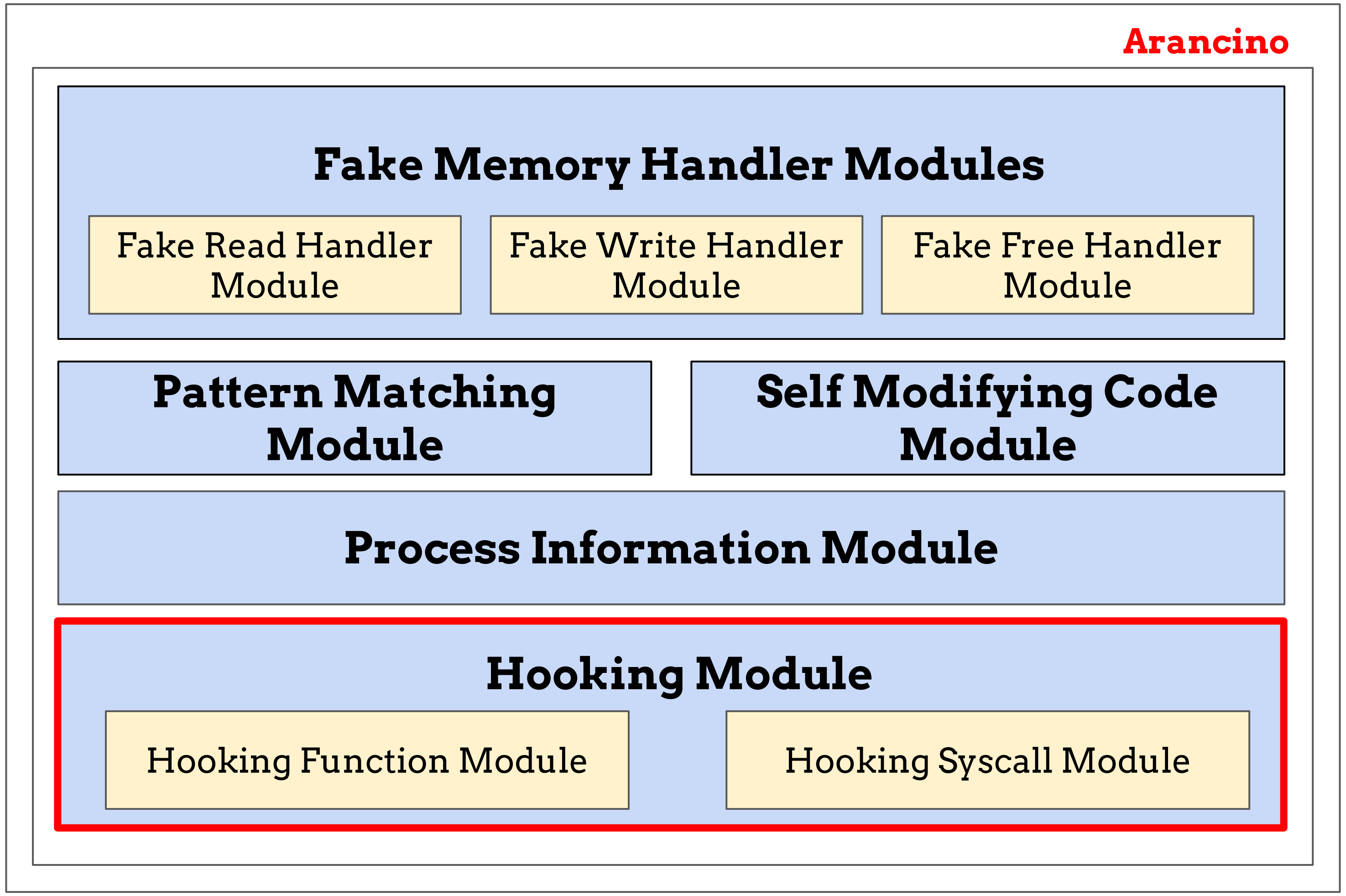
- NtQuerySystemInformation
- CSRSS.exe



Arancino - Hooking Module

- **Hooking Function Module**: Install an Hook on dll's Functions

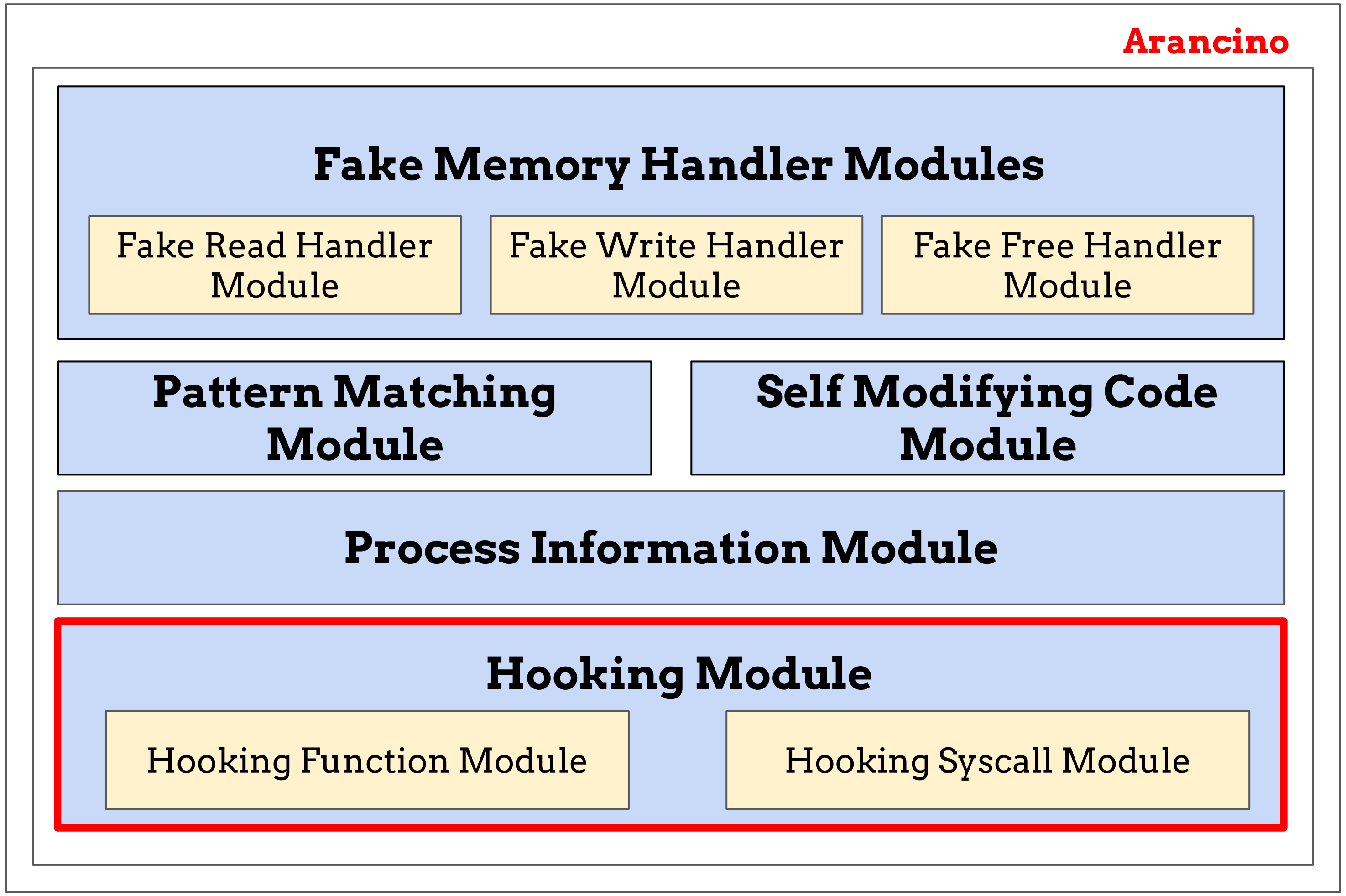
- **Hooking Syscall Module**: Install an Hook on dll's Functions



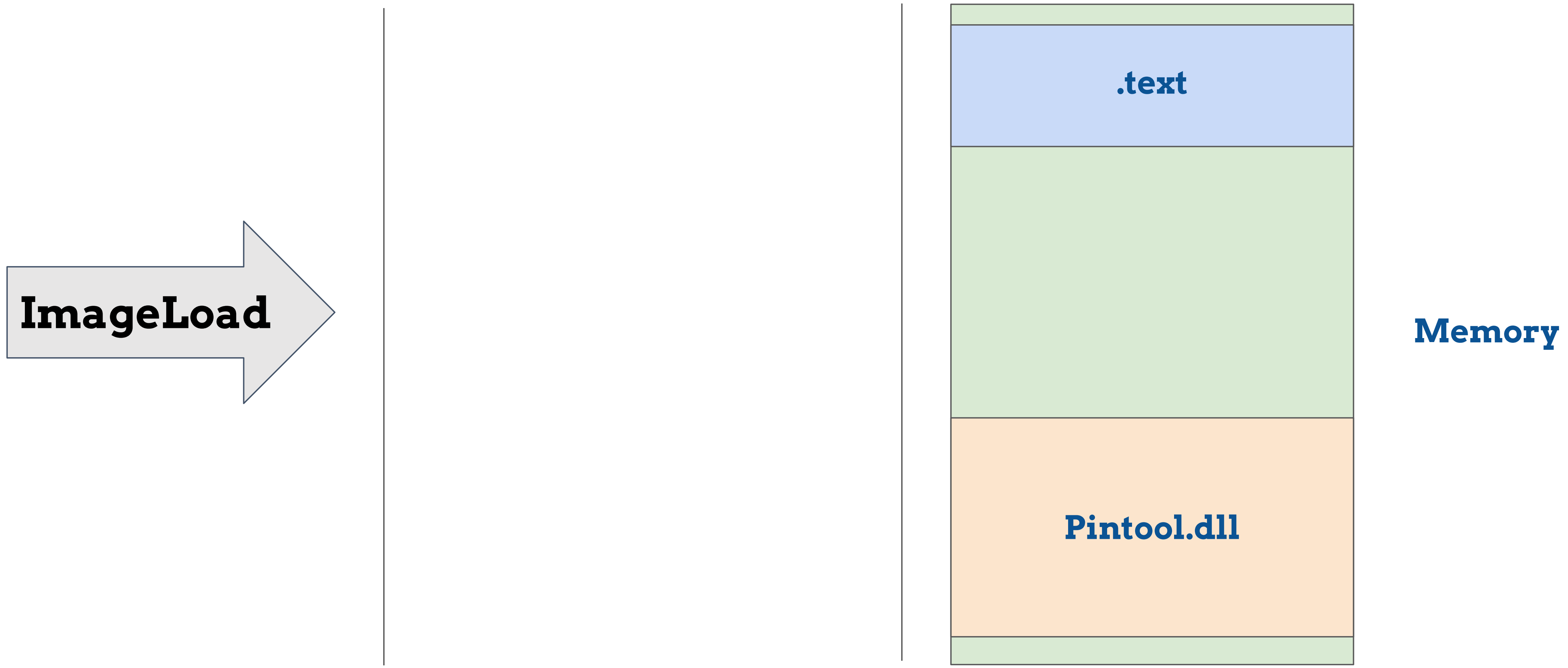
Arancino - Hooking Module

- **Hooking Function Module**: Install an Hook on dll's Functions

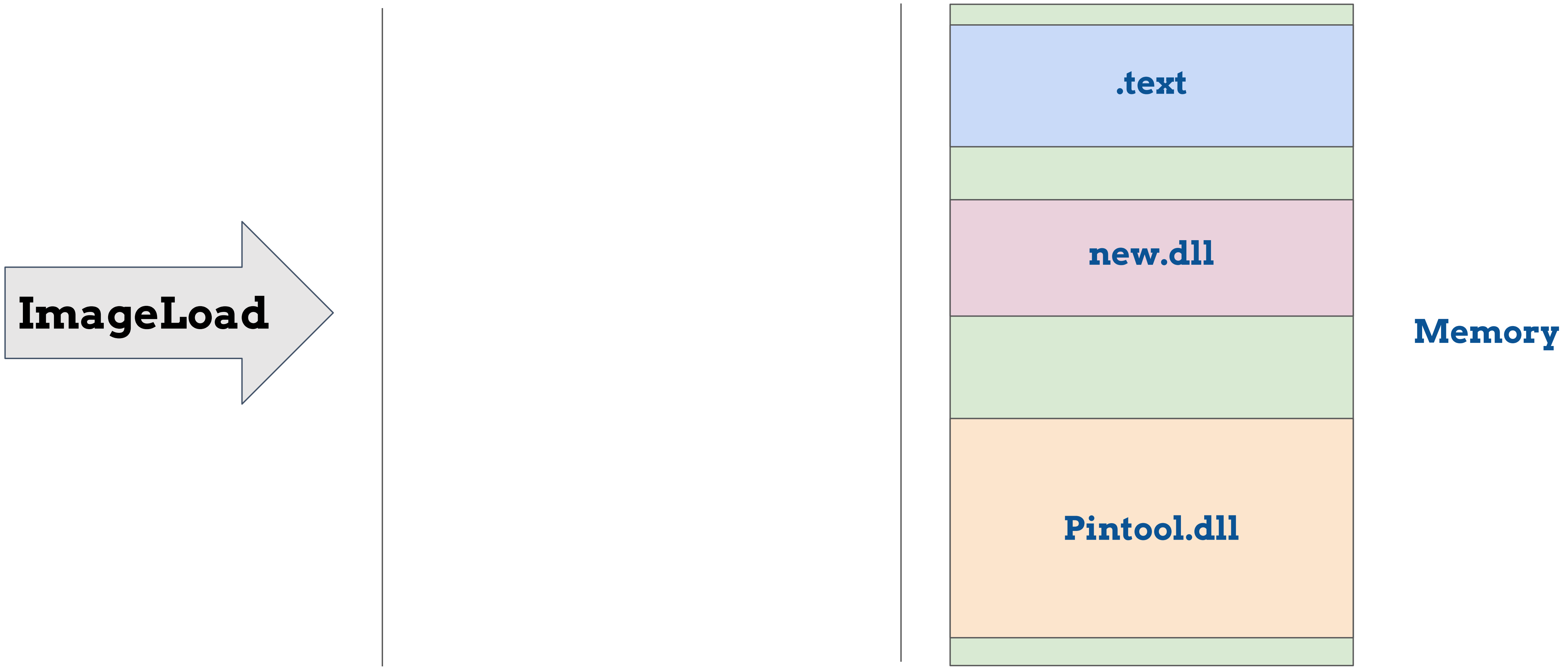
- **Hooking Syscall Module**: Install an Hook on dll's Functions



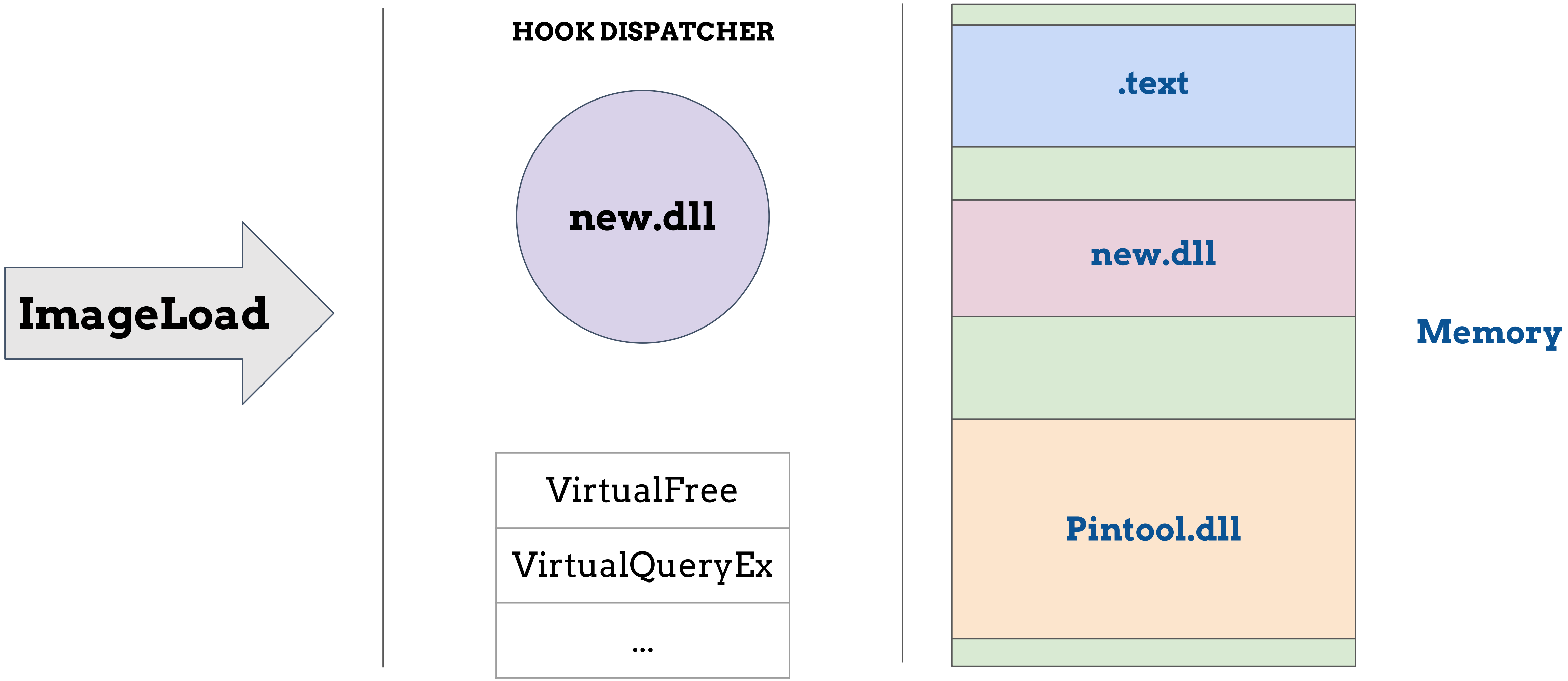
Arancino - Hook Functions



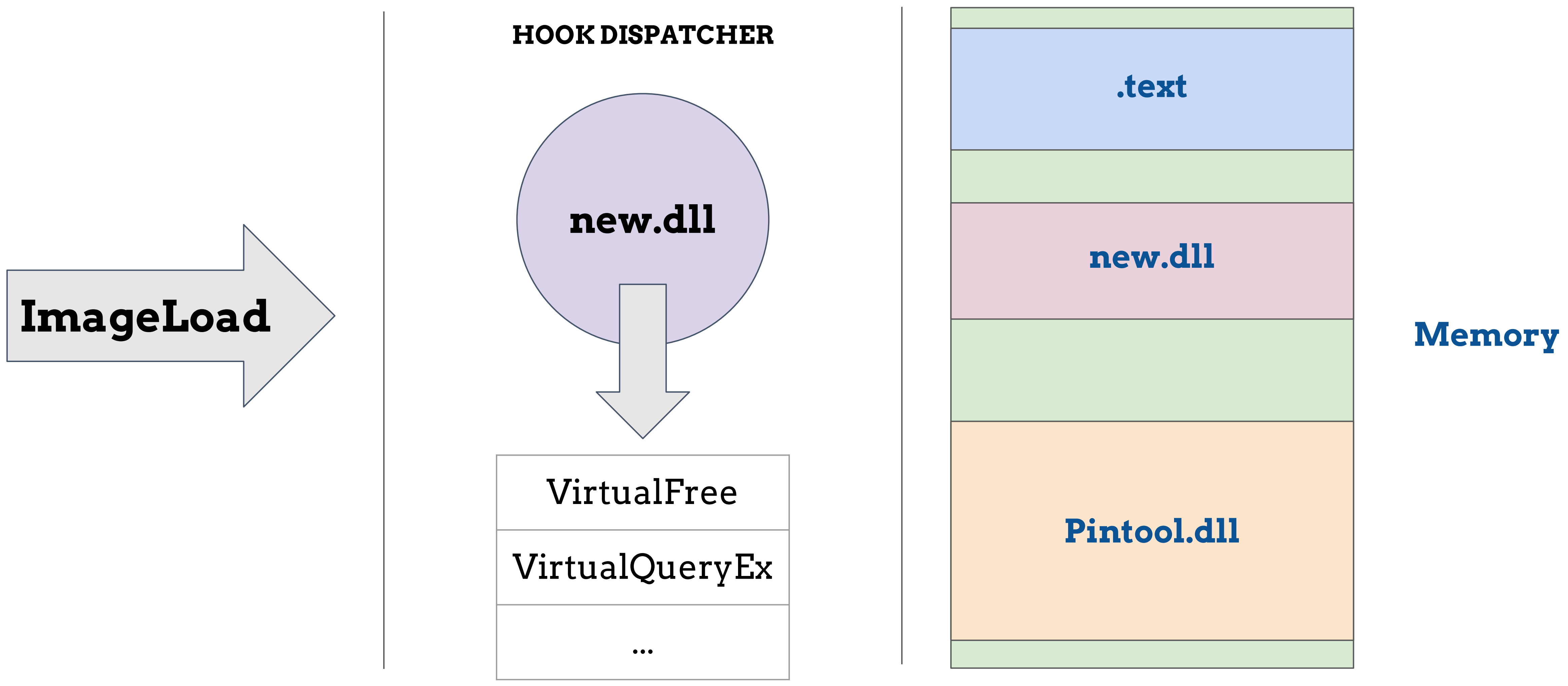
Arancino - Hook Functions



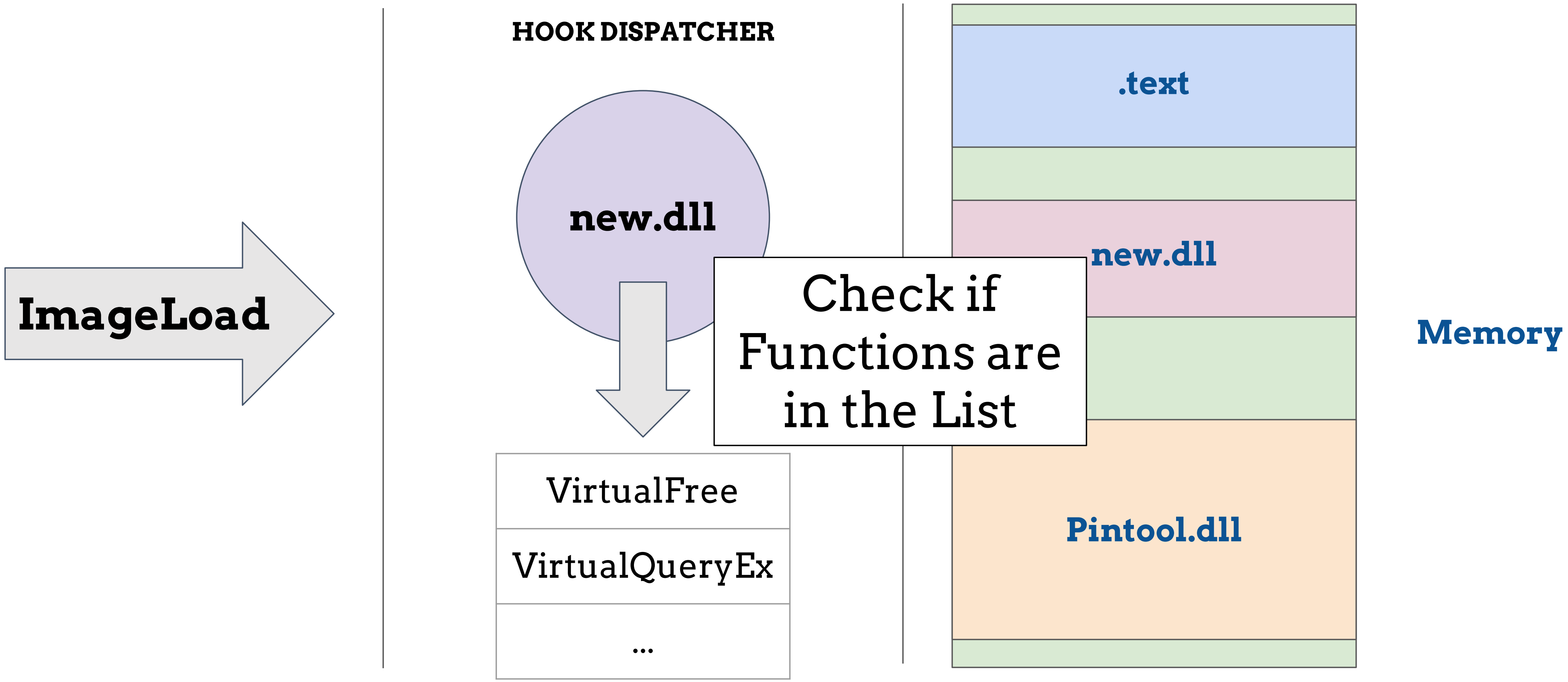
Arancino - Hook Functions



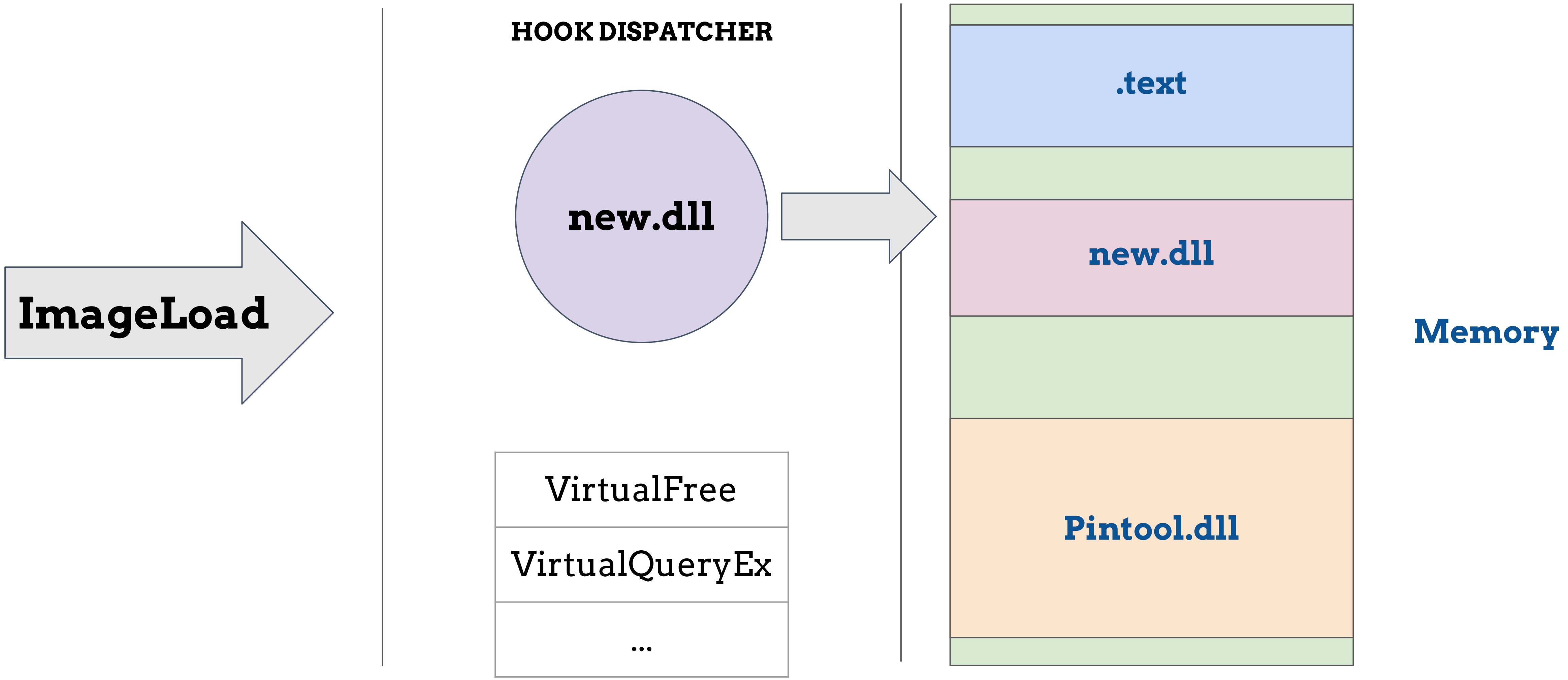
Arancino - Hook Functions



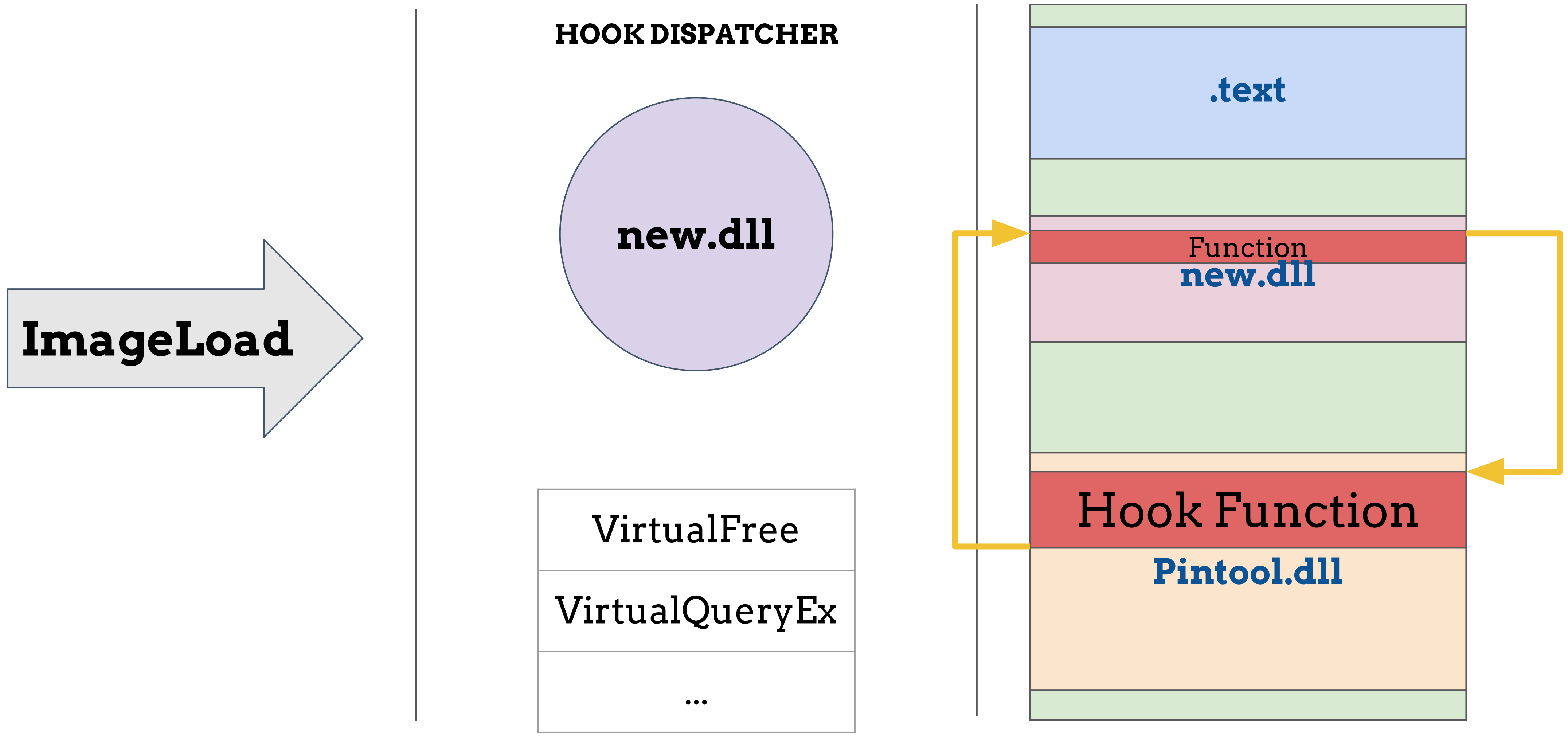
Arancino - Hook Functions



Arancino - Hook Functions



Arancino - Hook Functions



EA - Parent Detection

Hooked NtQuerySystemInformation

```
pin.exe -> cmd.exe
```

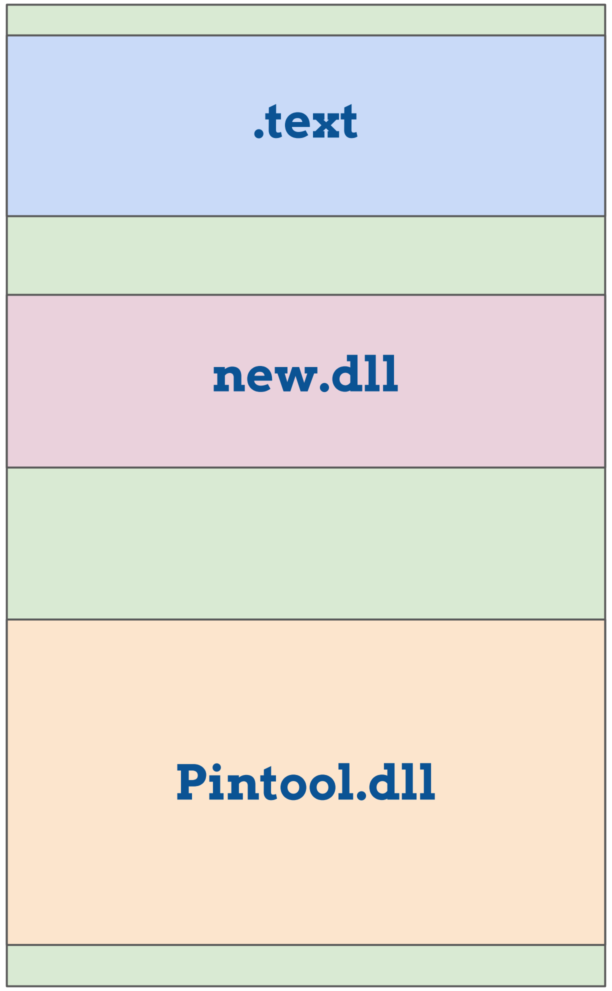
Hooked NtOpenProcess

to deny access to `CSRSS.exe`

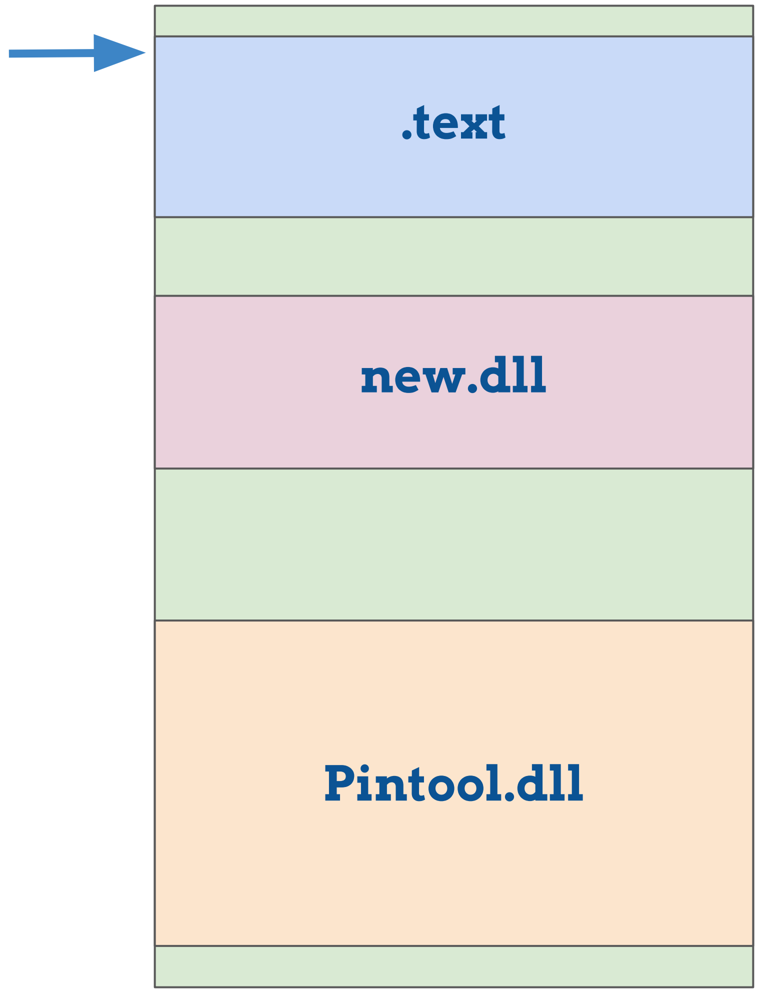
Environment Artifacts

- **Parent Detection**
- **Memory Fingerprinting**

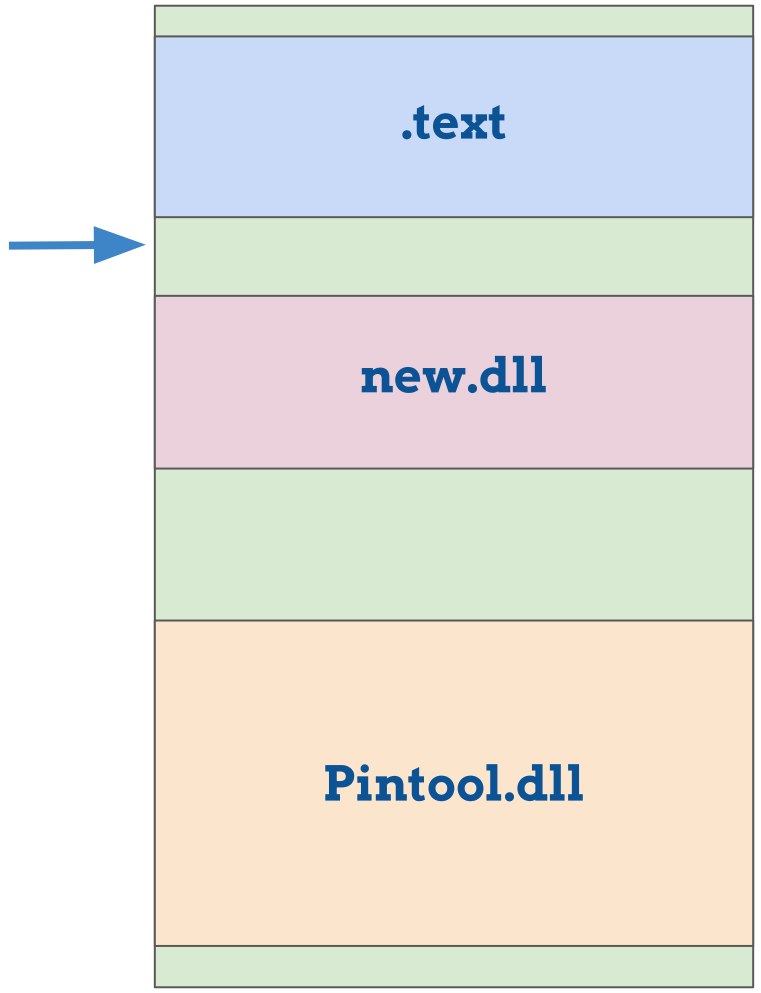
EA - Memory Fingerprinting



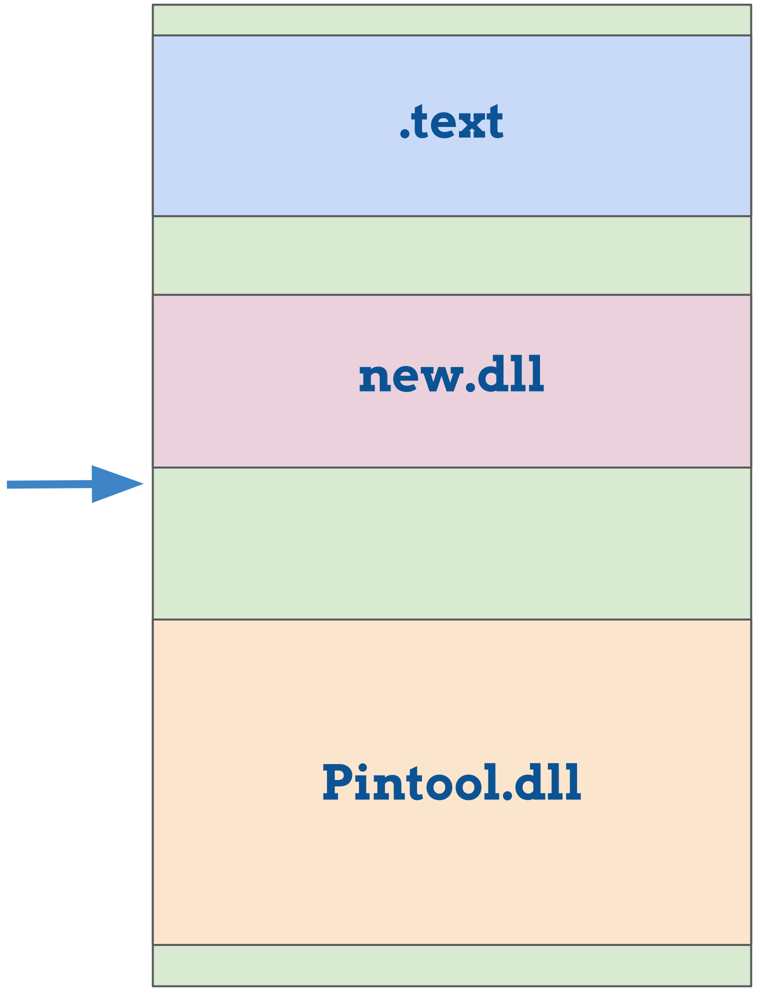
EA - Memory Fingerprinting



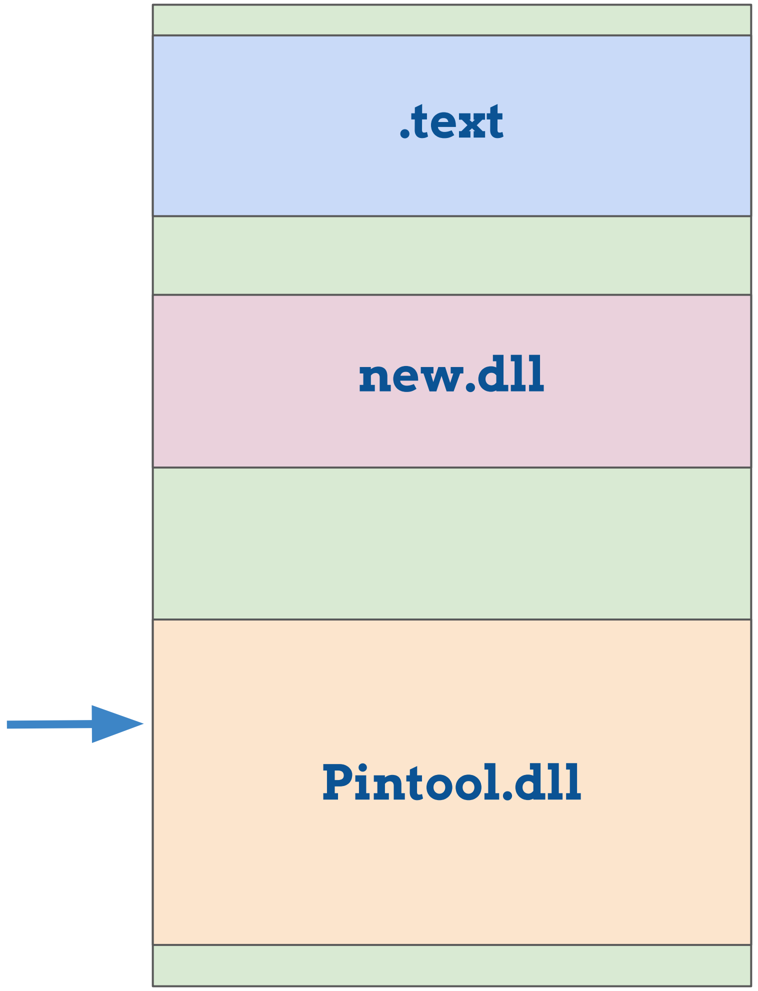
EA - Memory Fingerprinting



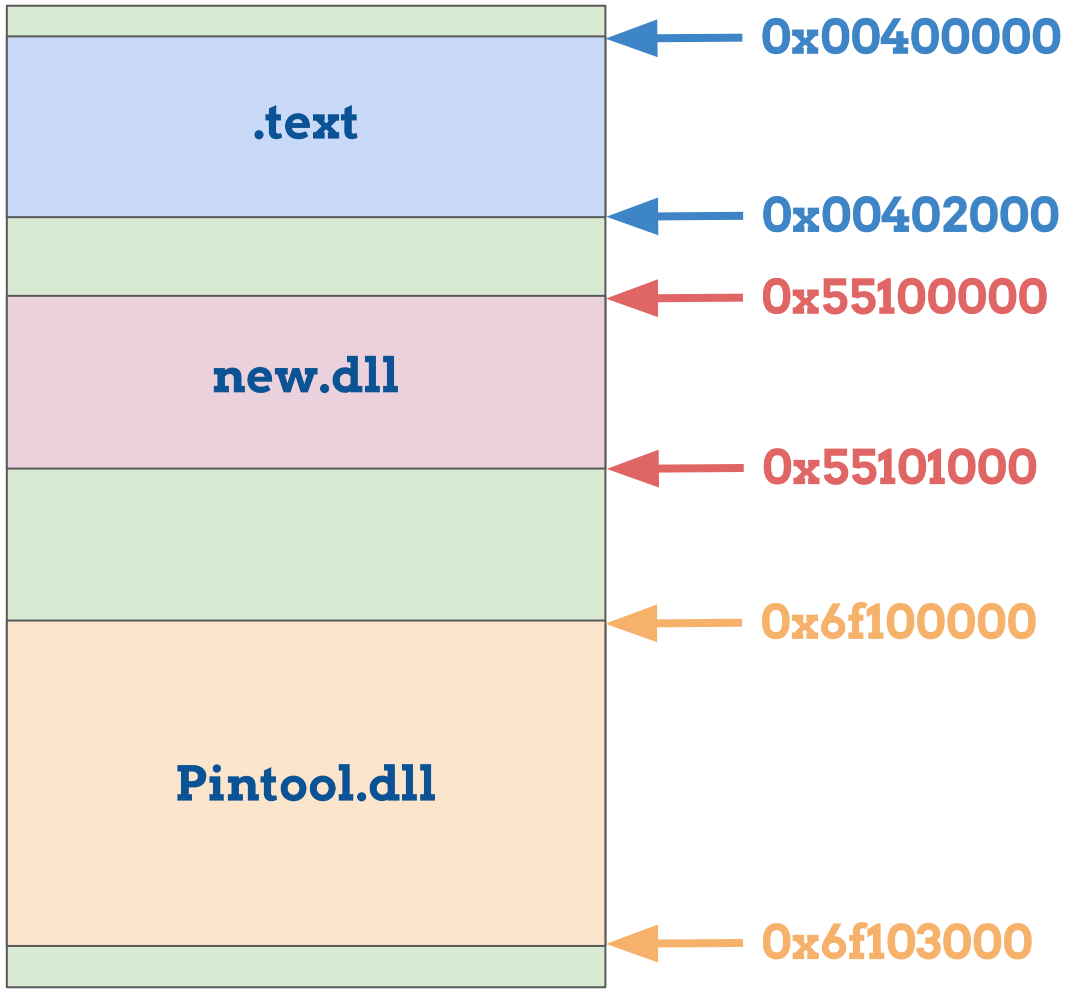
EA - Memory Fingerprinting



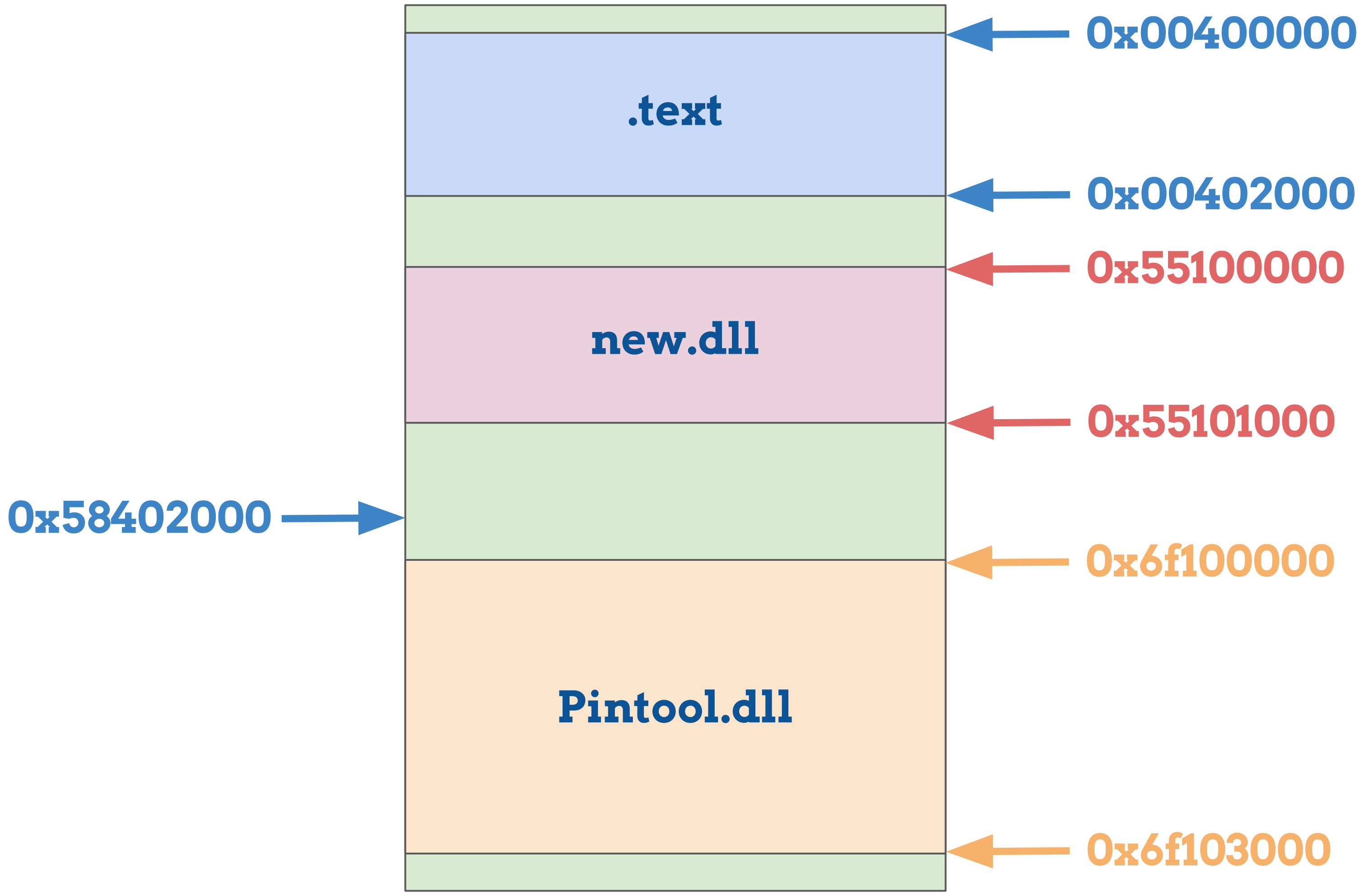
EA - Memory Fingerprinting



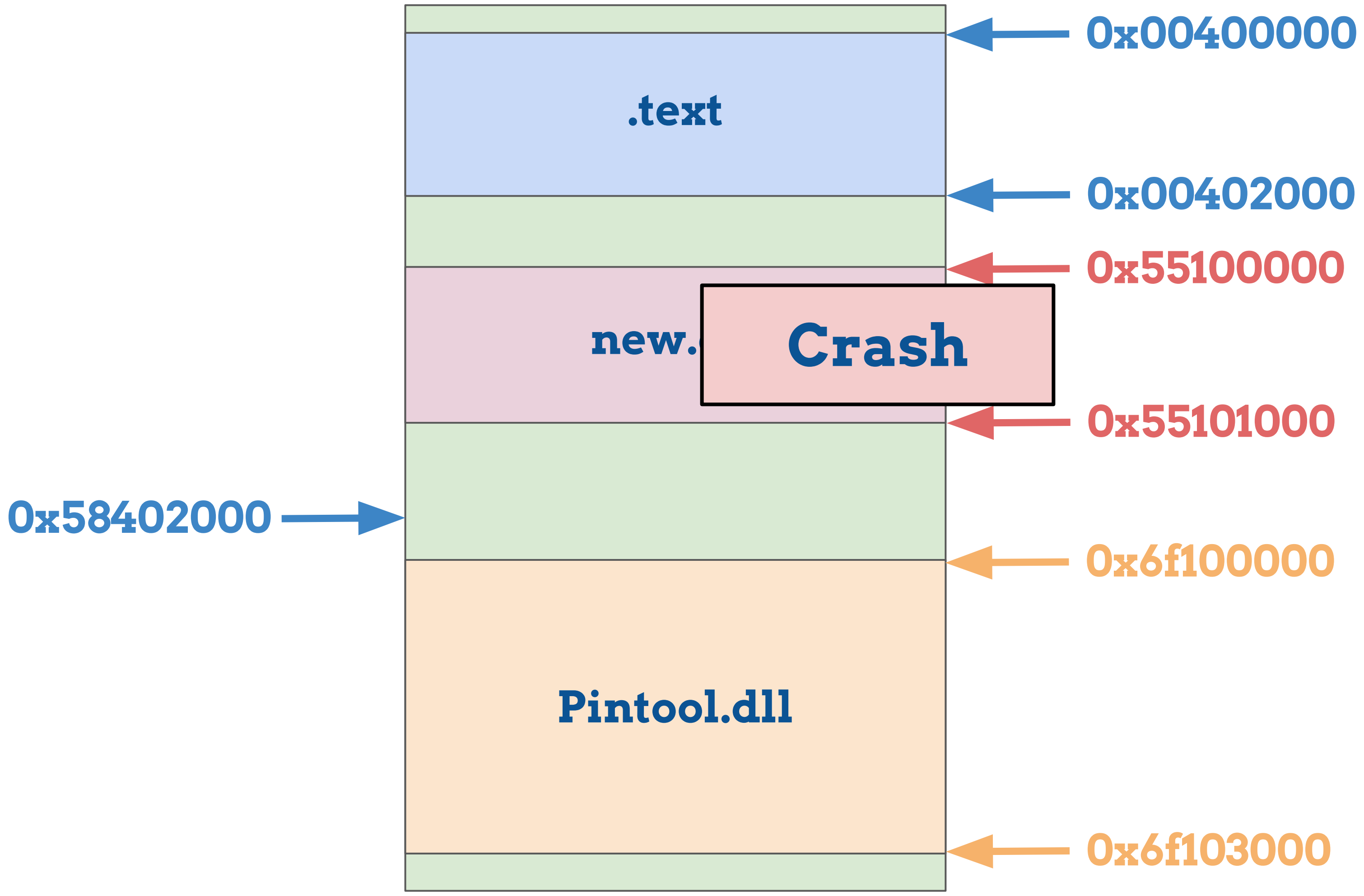
EA - Memory Fingerprinting



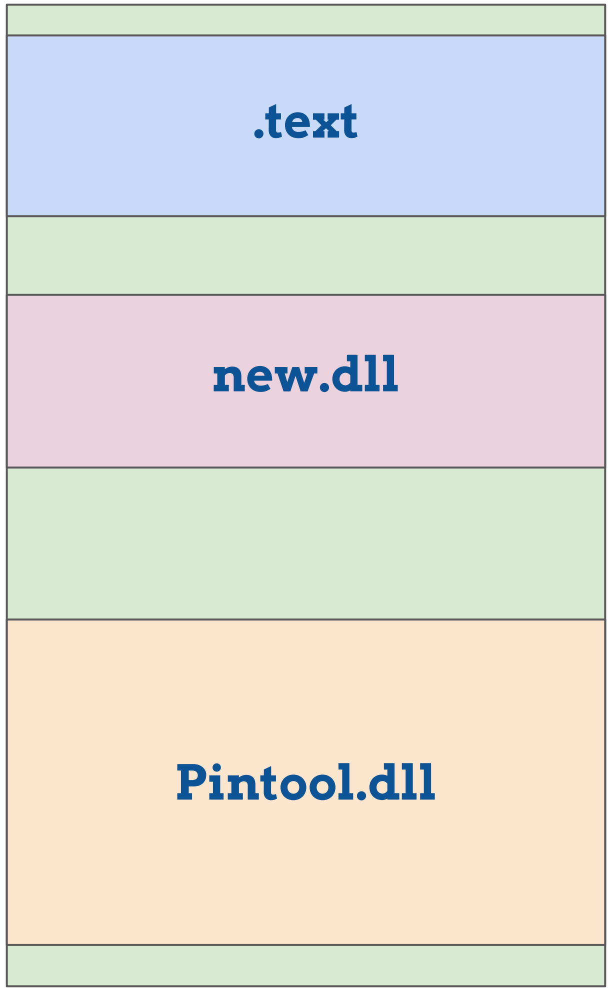
EA - Memory Fingerprinting



EA - Memory Fingerprinting



EA - Memory Fingerprinting



VirtualQuery

EA - Memory Fingerprinting

We Hook **NtQueryVirtualMemory**

We create a **Whitelist** of accessible memory regions updated at runtime.

- **Main Module**
- **Libraries**
- **Heap and Stack**
- **PEB, TEB, etc.**
- **Mapped files**

JIT Compiler Detection

JIT Compiler Detection

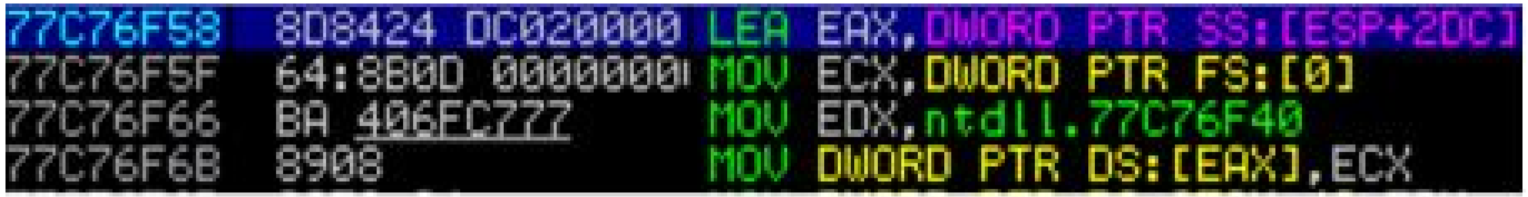
- **Memory Page Permissions**
  - Checks if there are **WX pages**
- **DLL Hook Detection**
- **Memory Allocations**

JIT Compiler Detection

- **Memory Page Permissions**
  - Checks if there are **WX pages**
- **DLL Hook Detection**
- **Memory Allocations**

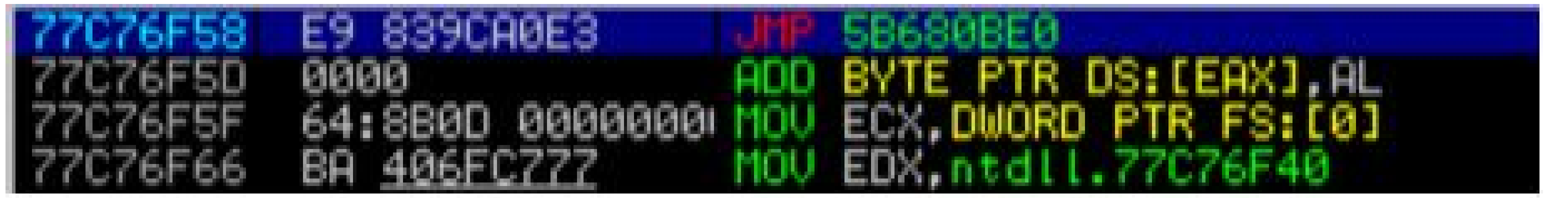
JITC Detection - DLL Hook

A process can search through memory for discrepancy caused by Hooks.



```
77C76F58   8D8424 DC020000 LEA EAX,DWORD PTR SS:[ESP+2DC]
77C76F5F   64:8B0D 0000000 MOV ECX,DWORD PTR FS:[0]
77C76F66   BA 406FC777     MOV EDX,ntdll.77C76F40
77C76F6B   8908            MOV DWORD PTR DS:[EAX],ECX
```

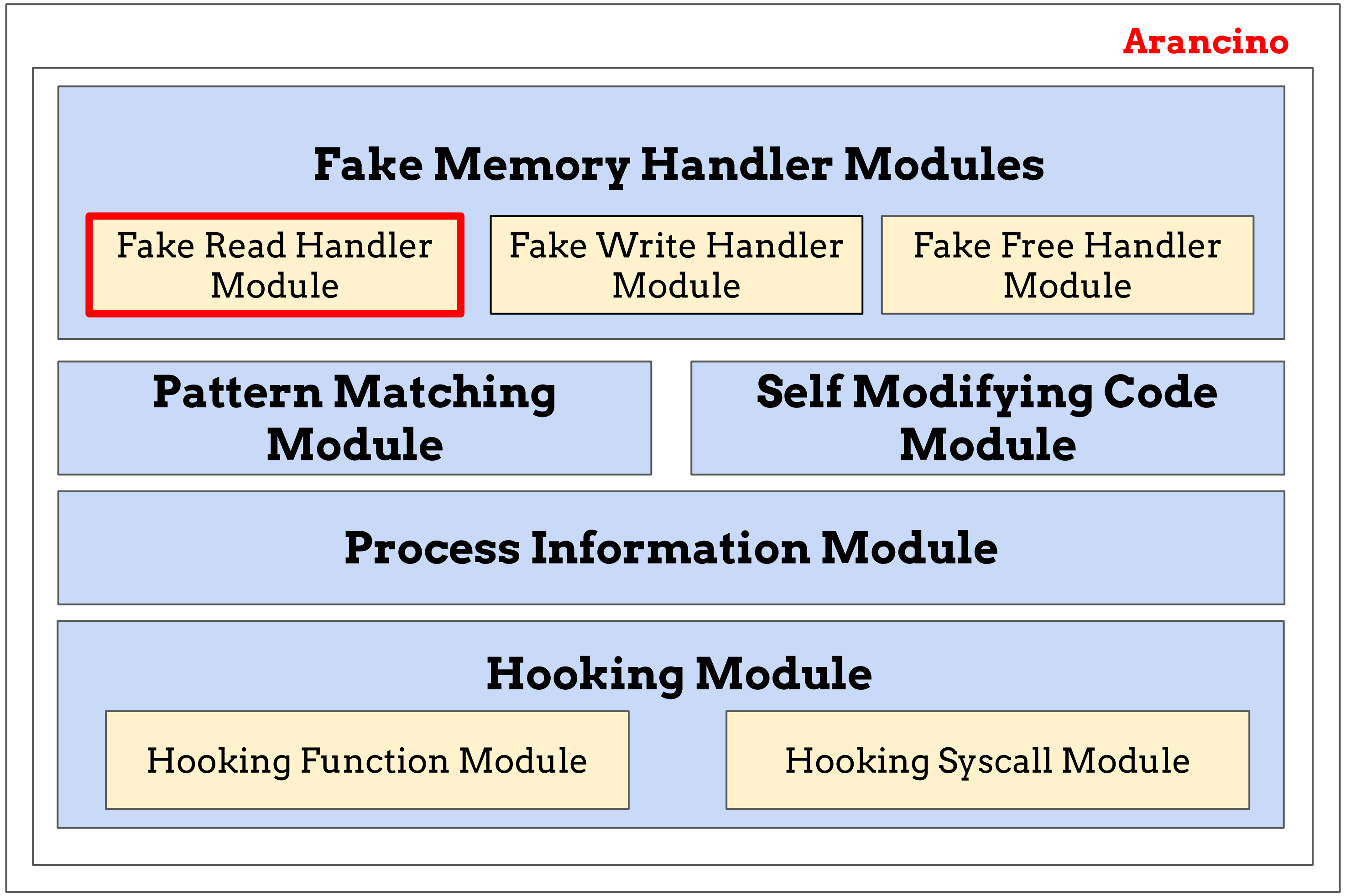
`KiUserApcDispatcher` - **normal execution**



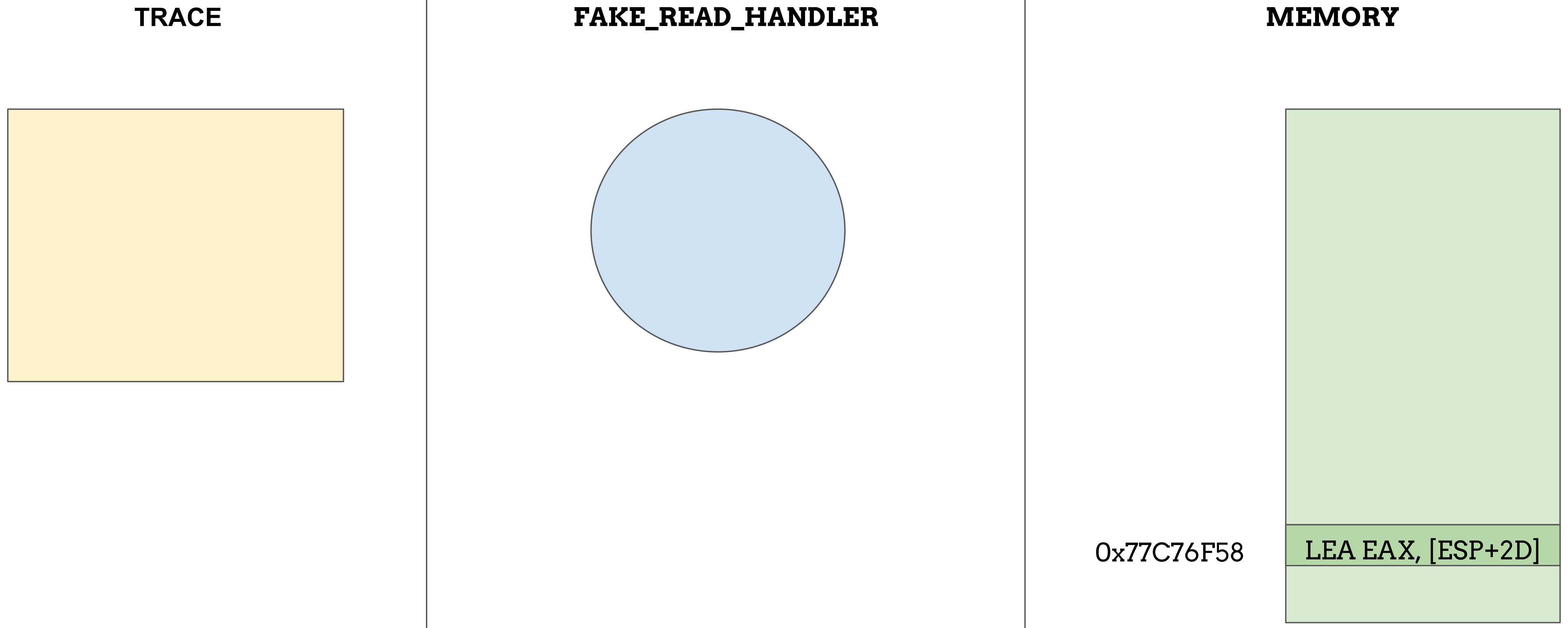
```
77C76F58   E9 839CA0E3     JMP 5B680BE0
77C76F5D   0000            ADD BYTE PTR DS:[EAX],AL
77C76F5F   64:8B0D 0000000 MOV ECX,DWORD PTR FS:[0]
77C76F66   BA 406FC777     MOV EDX,ntdll.77C76F40
```

`KiUserApcDispatcher` - **Instrumented execution**

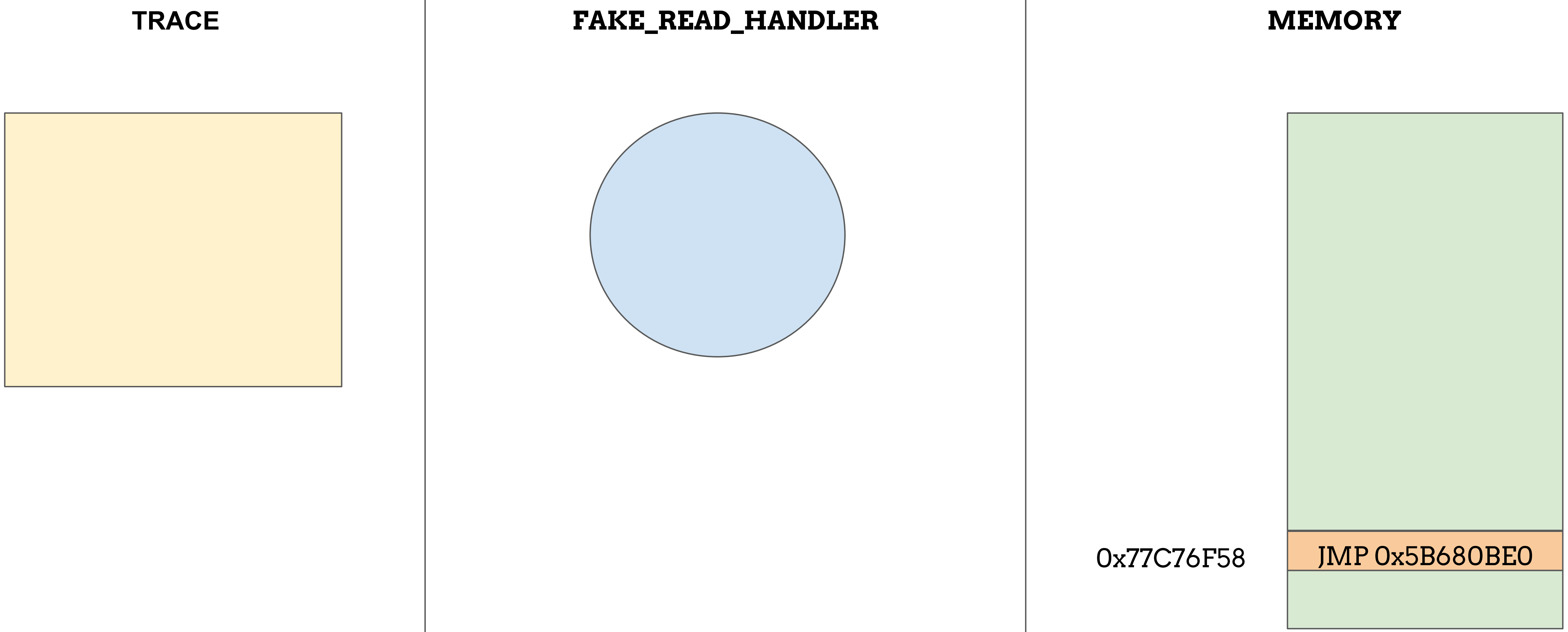
Arancino


Arancino
Fake Memory Handler Modules
Fake Read Handler Module
Fake Write Handler Module
Fake Free Handler Module
Pattern Matching Module
Self Modifying Code Module
Process Information Module
Hooking Module
Hooking Function Module
Hooking Syscall Module

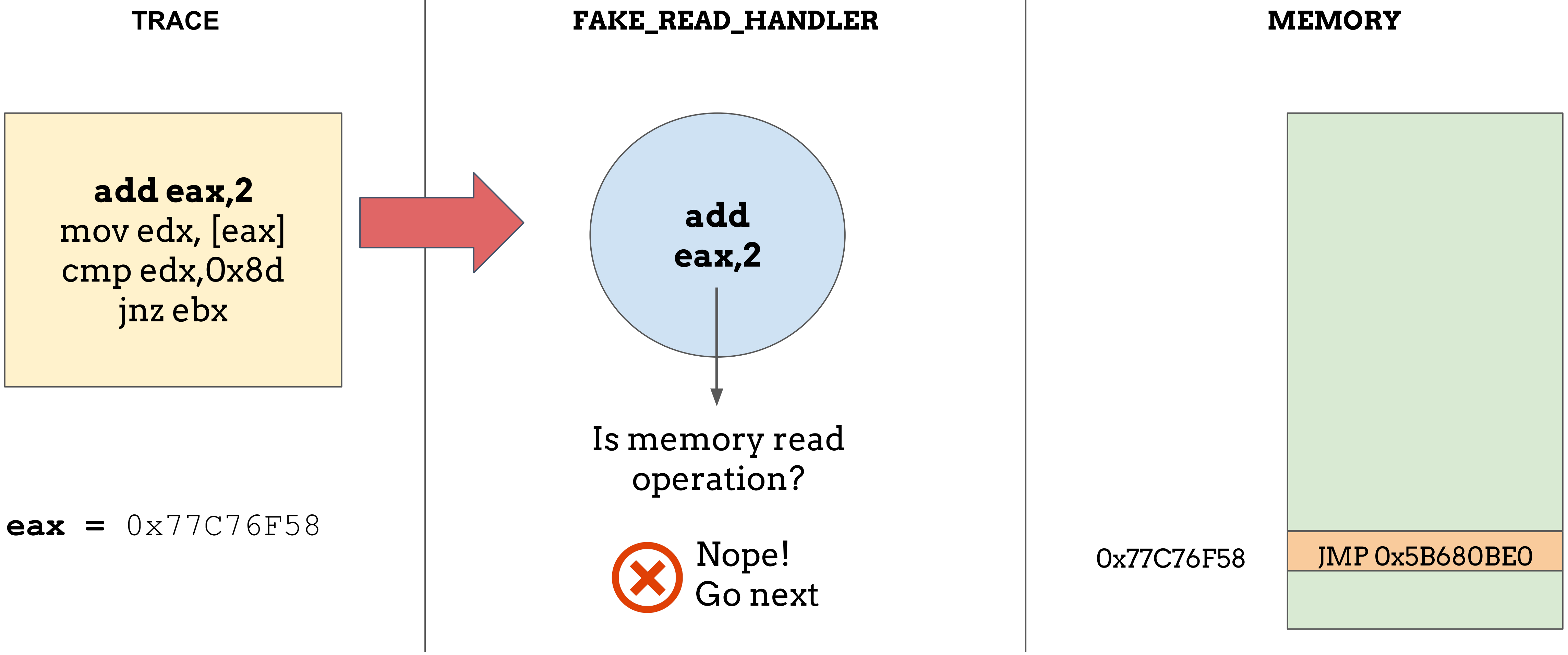
JITC Detection - DLL Hook



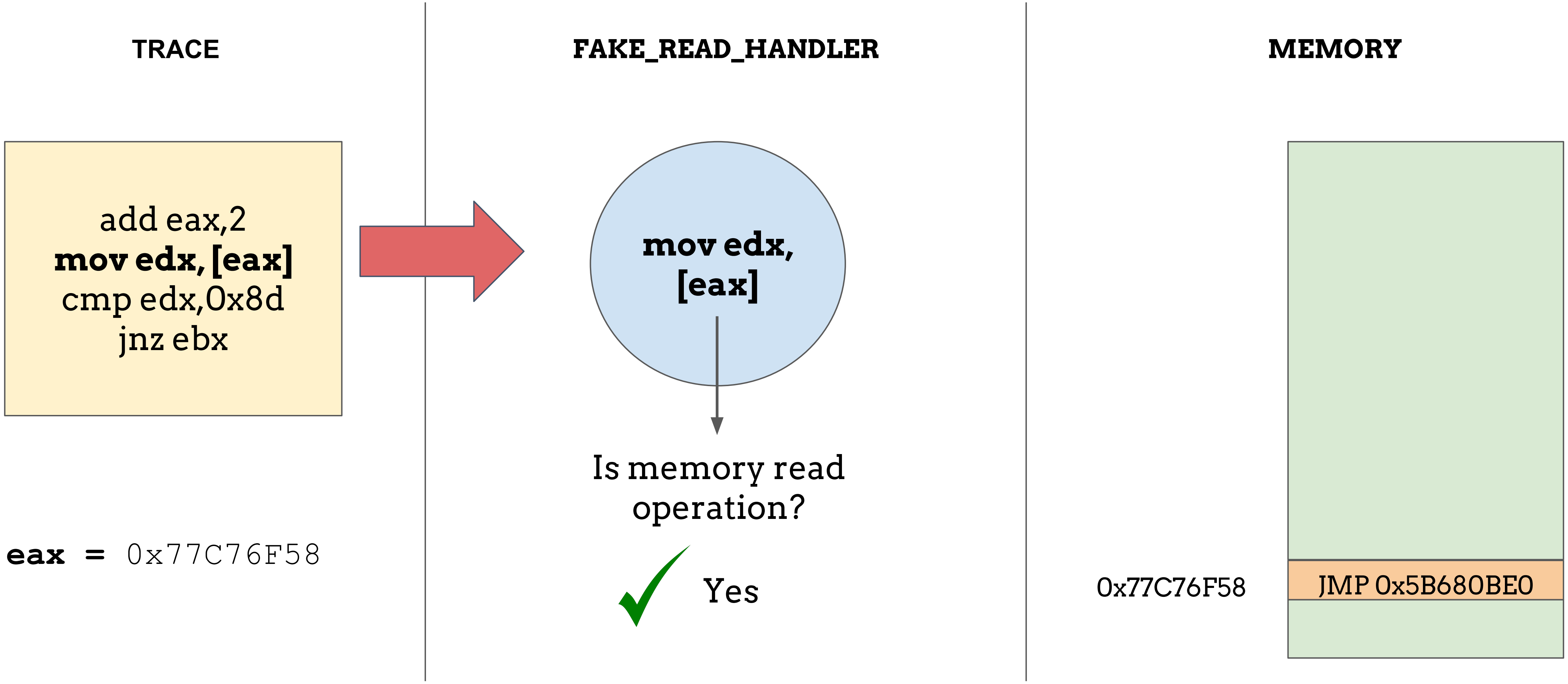
JITC Detection - DLL Hook



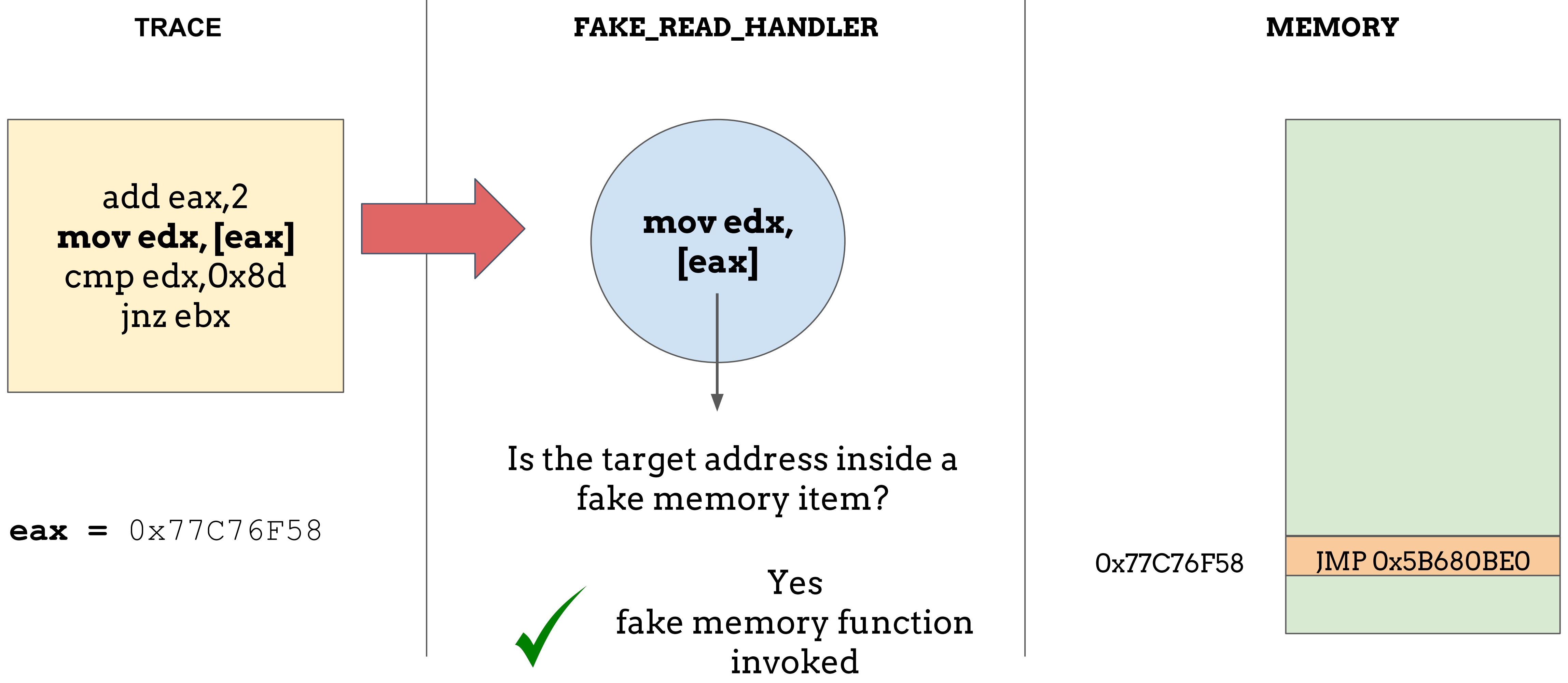
JITC Detection - DLL Hook



JITC Detection - DDL Hook



JITC Detection - DDL Hook



TRACE

add eax,2
mov edx, [eax]
cmp edx,0x8d
jnz ebx

`eax = 0x77C76F58`

FAKE_READ_HANDLER

mov edx, [eax]

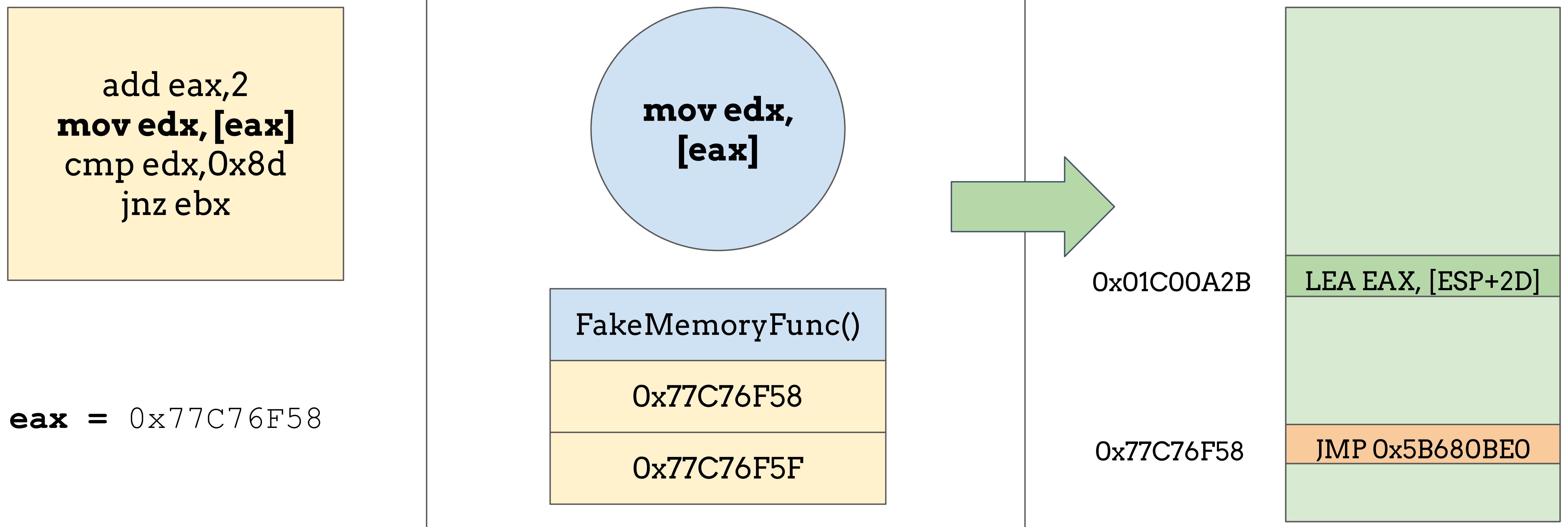
Is the target address inside a fake memory item?

✓ Yes
fake memory function invoked

MEMORY

0x77C76F58 | JMP 0x5B680BE0

JITC Detection - DDL Hook



TRACE

```
add eax,2
mov edx, [eax]
cmp edx,0x8d
jnz ebx
```

eax = 0x77C76F58

FAKE_READ_HANDLER

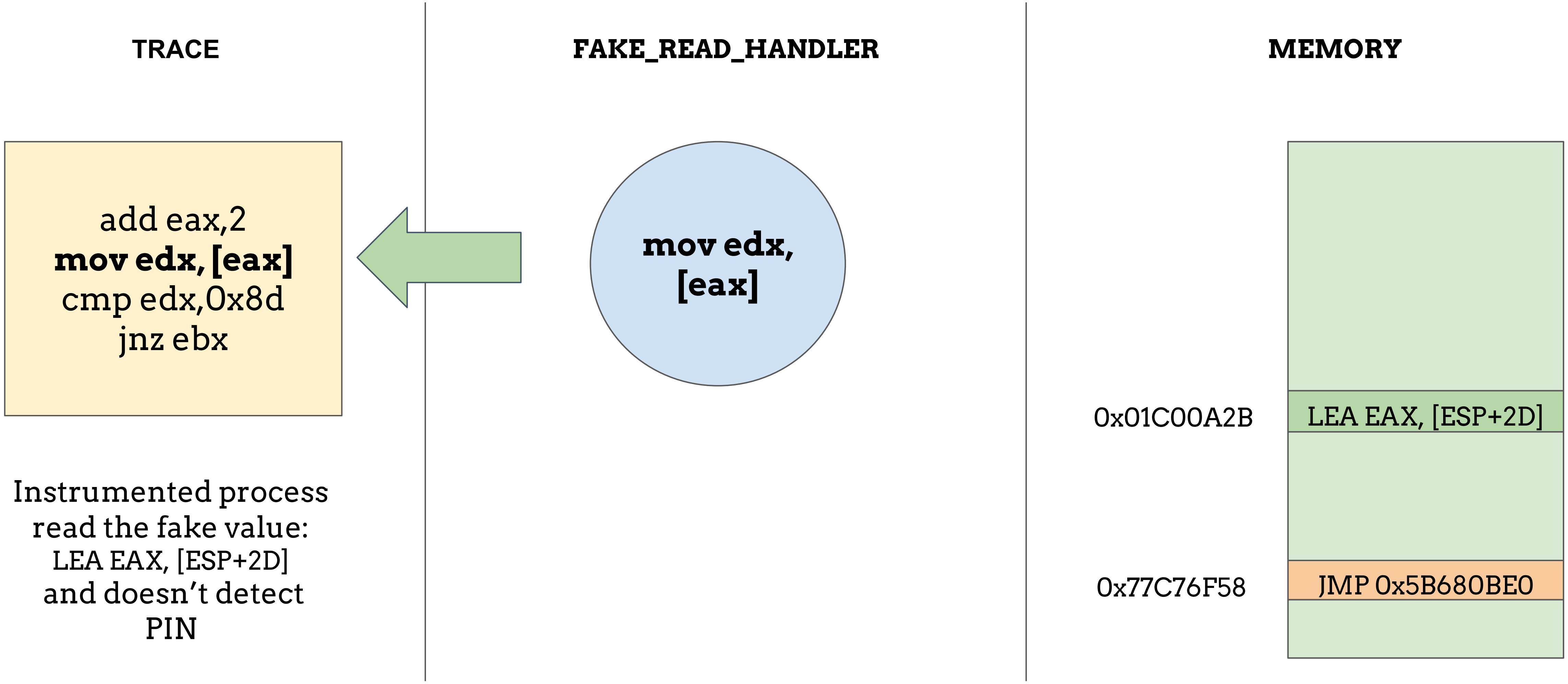
```
mov edx,
[eax]
```

FakeMemoryFunc()
0x77C76F58
0x77C76F5F

MEMORY

0x01C00A2B	LEA EAX, [ESP+2D]
0x77C76F58	JMP 0x5B680BE0

JITC Detection - DDL Hook



Instrumented process read the fake value: LEA EAX, [ESP+2D] and doesn't detect PIN

JIT Compiler Detection

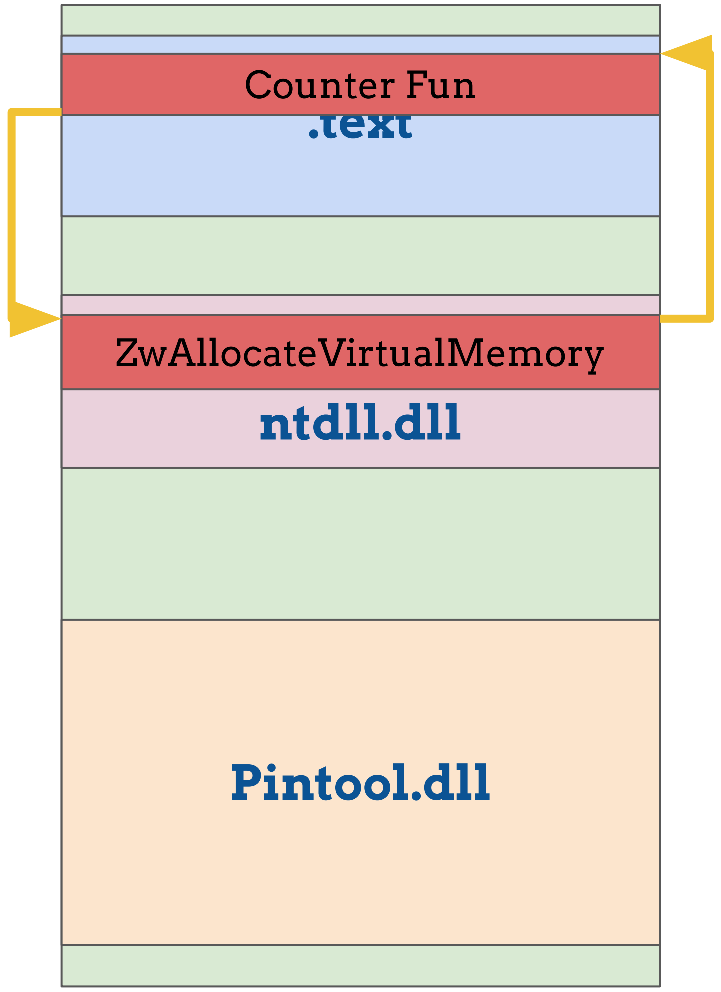
- **Memory Page Permissions**
  - Checks if there are **WX pages**
- **DLL Hook Detection**
- **Memory Allocations**

JIT Compiler - API Hook

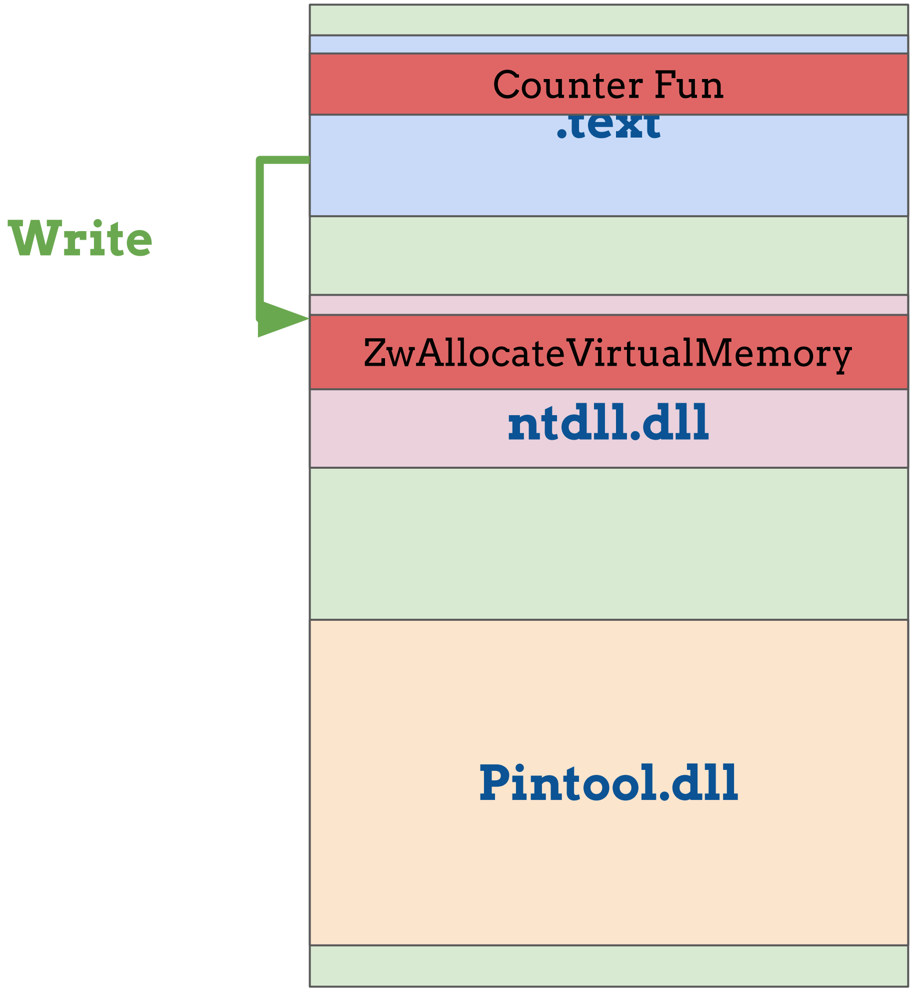
JIT Compiler needs **Memory** to perform the compiling

We can monitor the allocation by Hooking at **ZwAllocateVirtualMemory**

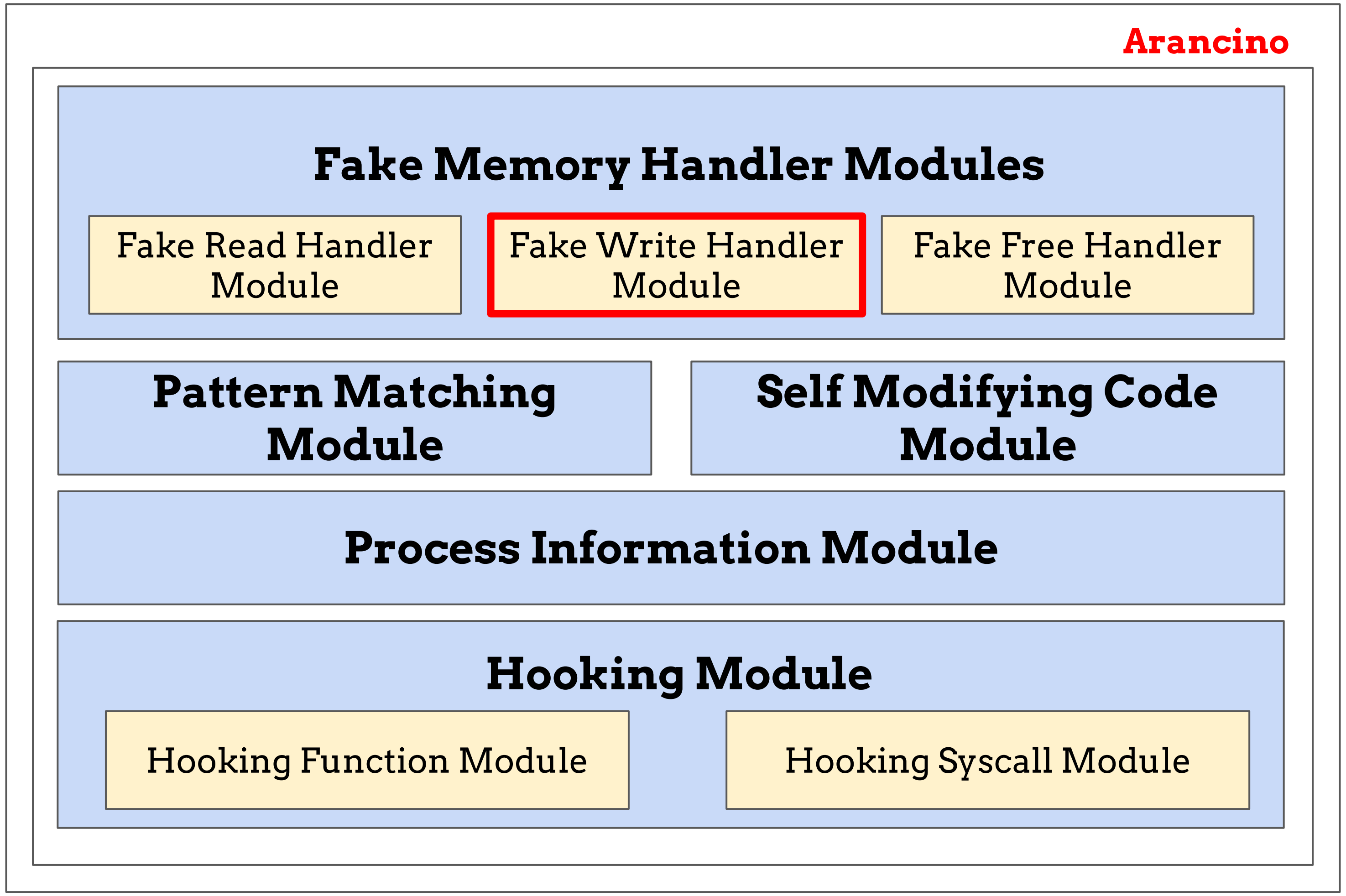
JIT Compiler - API Hook



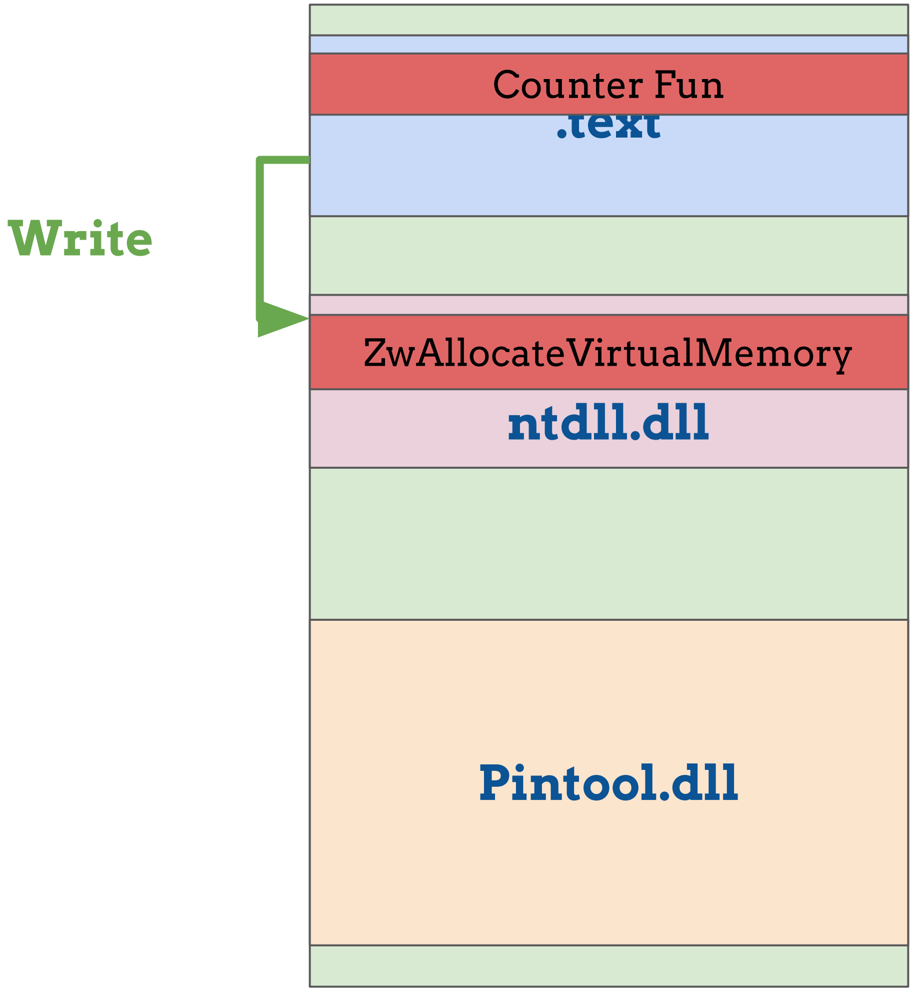
JIT Compiler - API Hook



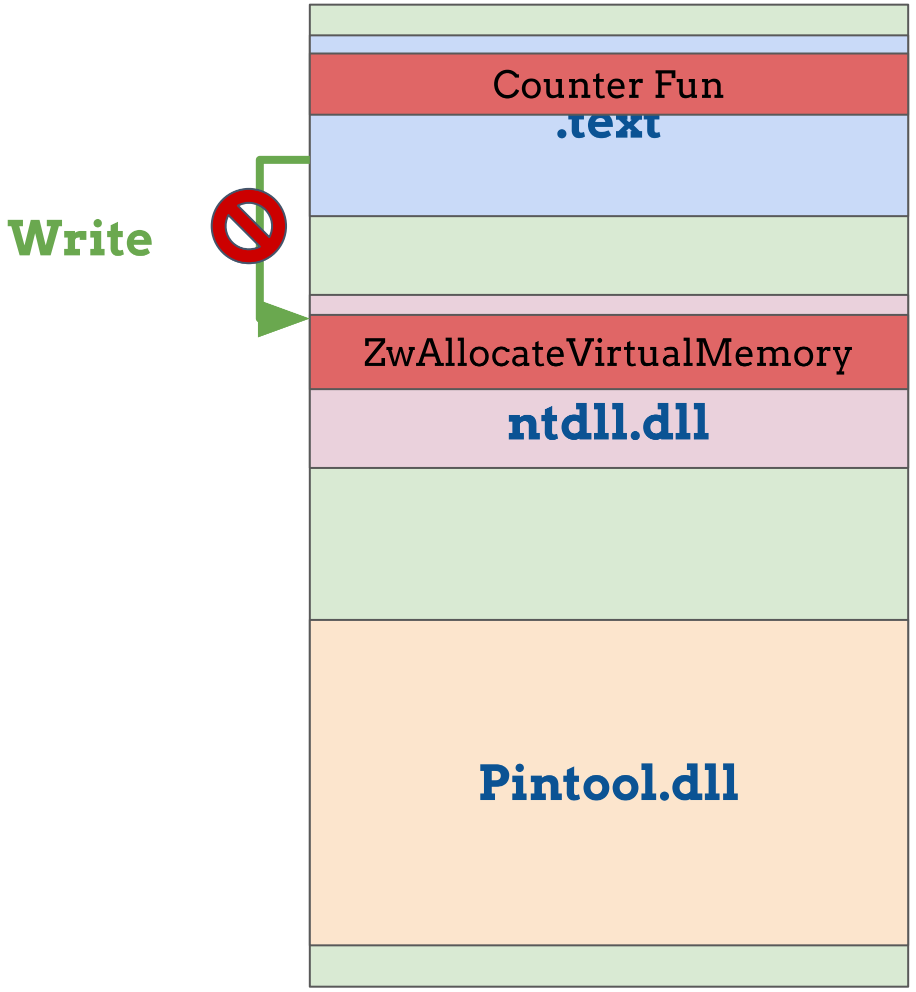
Arancino


Arancino
Fake Memory Handler Modules
Fake Read Handler Module
Fake Write Handler Module
Fake Free Handler Module
Pattern Matching Module
Self Modifying Code Module
Process Information Module
Hooking Module
Hooking Function Module
Hooking Syscall Module

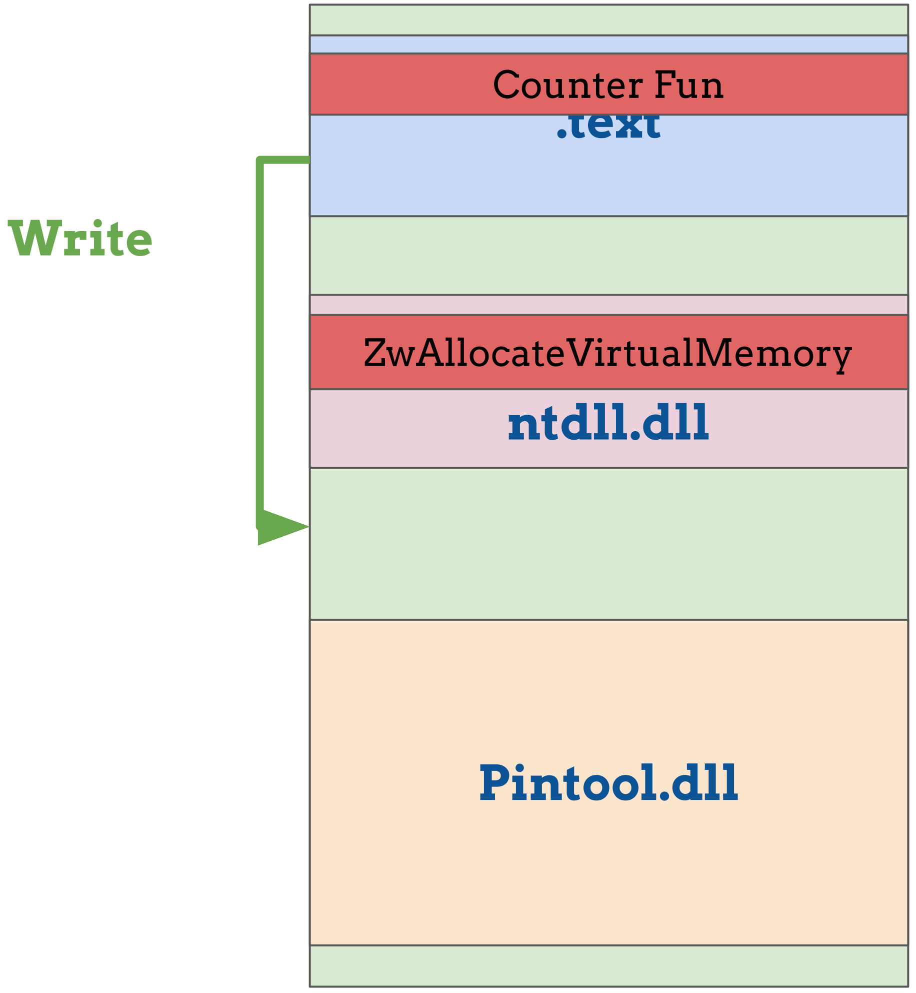
JIT Compiler - API Hook



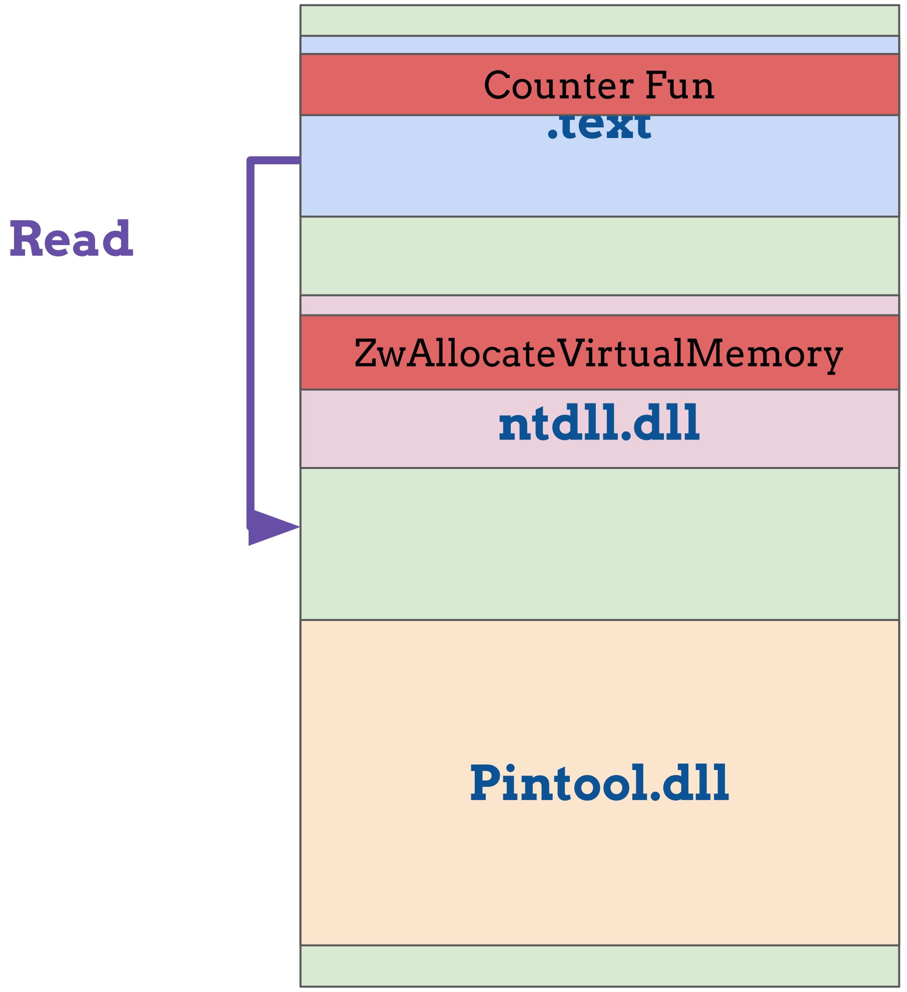
JIT Compiler - API Hook



JIT Compiler - API Hook



JIT Compiler - API Hook



Overhead Detection

Overhead Detection

- **Windows Time**
  - Use windows API
    - GetTickCount and timeGetTime
  - Or Windows Structures
    - KUSER_SHARED_DATA.

- **CPU Time**
  - Count CPU cycles (`rdtsc`)

Evasive Malware Measurement

Anti-Instrumentation Measurement

Dataset

- **7006** Binaries
- Virus Total Intelligence (3+ AV Detection)
- From October 2016 to February 2017

Anti-Instrumentation Measurement

Environment Setup

- **Virtual Machine** (VirtualBox)
- **Windows 7** (64-bit)
- **Custom Apps** (Adobe Reader, Chrome, and media players)
- **User Data** (saved credentials, browser history, etc.)
- **Basic User Activity** (moving the mouse, launching applications)
- **5 min** run

Evasive Malware

At least one evasive behavior: 1,093 / 7006 (15.6%)

Family Name [1]	Samples	Evasive	Techniques
virlock	619 (8.8%)	600 (96.9%)	2
confidence	505 (7.2%)	68 (13.5%)	4
virut	242 (3.5%)	13 (5.4%)	2
mira	230 (3.3%)	9 (3.9%)	1
upatre	187 (2.7%)	2 (1.1%)	1
lamer	171 (2.4%)	0 (0.0%)	0
sivis	168 (2.4%)	0 (0.0%)	0

[1] **AvClass** https://github.com/malicialab/avclass

black hat EUROPE 2017

Top Evasive Malware

At least one evasive behavior: 1,093 / 7006 (15.6%)

Family Name [1]	Samples	Evasive	Techniques
sfone	19	19 (100.0%)	1
unruy	11	11 (100.0%)	1
virlock	619	600 (96.9%)	2
vilsel	13	8 (61.5%)	2
urelas	18	9 (47.4%)	2
confuser	52	8 (44.4%)	1
vobfus	29	19 (36.5%)	1

[1] **AvClass** https://github.com/malicialab/avclass

Top Techniques Used

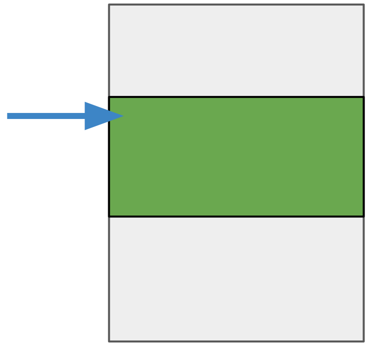
At least one evasive behavior: 1,093 / 7006 (15.6%)

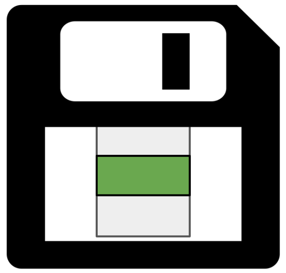
	Technique	#
Code Cache Artifacts	Self-modifying code	897
Environment Artifacts	Parent detection	259
JIT Compiler Detection	Write on protected memory region	40
Environment Artifacts	Check DEBUG flag	5
Environment Artifacts	Memory fingerprinting	3

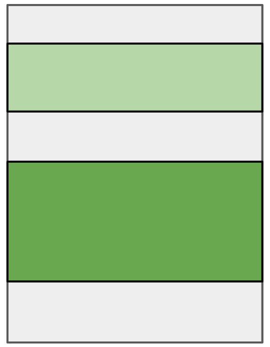
Overhead

	Pin time [ms]	Arancino [ms]	Arancino overhead [%]	Module activated
Parent Detection	850	870	2%	Hooking Module
EIP Detection - int2e	710	1,150	62%	Pattern Match Module
Memory Fingerprinting	2,000	7,090	254,5%	Fake Read Module
Memory Allocations	2,000	2,900	45%	Fake Write Module + Hooking Module

Unpacking Approach


Detect W and X memory regions


Dump the Program


Deobfuscate the Import Address Table

Recognize the correct dump

Experiment 1 : known packers

	Upx	FSG	Mew	mpress	PeCompact	Obsidium	ExePacker	ezip
MessageBox.exe	✓	✓	✓	✓	✓	!	✓	✓
WinRAR.exe	✓	✓	✓	✓	✓	!	✓	✓

	Xcomp	PElock	ASProtect	ASPack	eXpressor	exe32packer	beropacker	Hyperion
MessageBox.exe	✓	!	!	✓	!	✓	✓	✓
WinRAR.exe	✓	!	!	✓	!	✓	✓	✓

! → Original code dumped but Import directory not reconstructed

black hat EUROPE 2017

Experiment 2 : wild samples

Number of packed (checked manually) samples 1096

	N°	%
Unpacked and working	669	63
Unpacked but not executable	139	13
Not unpacked	258	24

DEMO Time!

Black Hat Sound Bytes

- Malware authors employ **Anti-Instrumentation** techniques to detect when their samples are being instrumented
- We proposed an approach to practically **defeat such techniques**
- We studied the **common techniques** adopted by modern malware authors to evade of instrumentation systems
- On top of Arancino ~> dynamic, **evasion-resilient unpacker**
    - Known packers use anti-instrumentation techniques!

Thanks!

https://github.com/necst/arancino

Mario Polino

<mario.polino@polimi.it>

Questions?

https://github.com/necst/arancino

Mario Polino

<mario.polino@polimi.it>

Questions?

https://github.com/necst/arancino

Mario Polino

<mario.polino@polimi.it>

Credits

- Icons, CC from Noun Project:
  - Vicons Design
  - Aya Sofya
  - Adnen Kadri
  - Stock Image Folio
  - Icon Fair
  - Creative Stall
  - Gregor Cresnar