

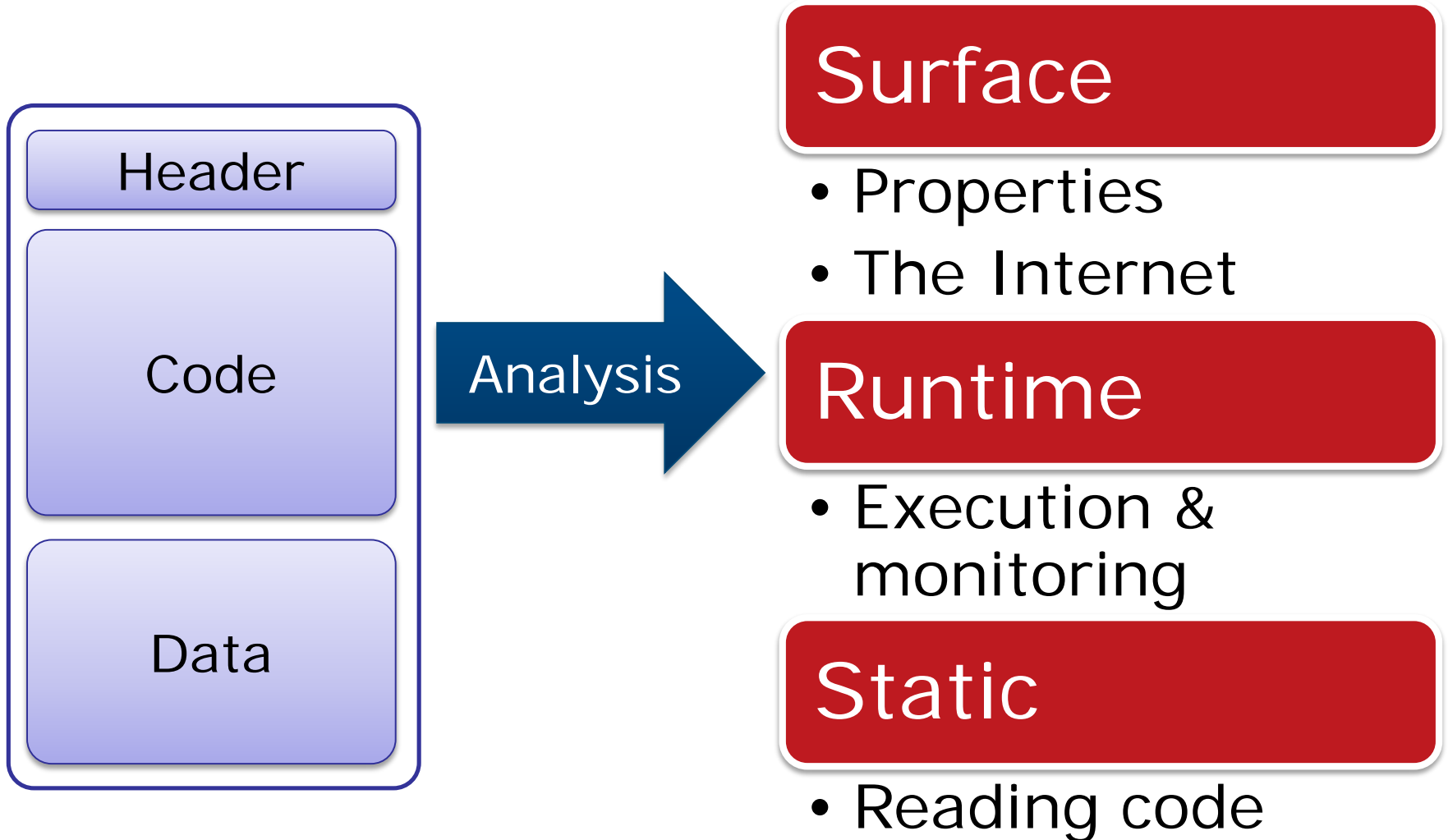
Recommendation of Perfect Unpacking

2014/04/24

JPCERT/CC Analysis Center

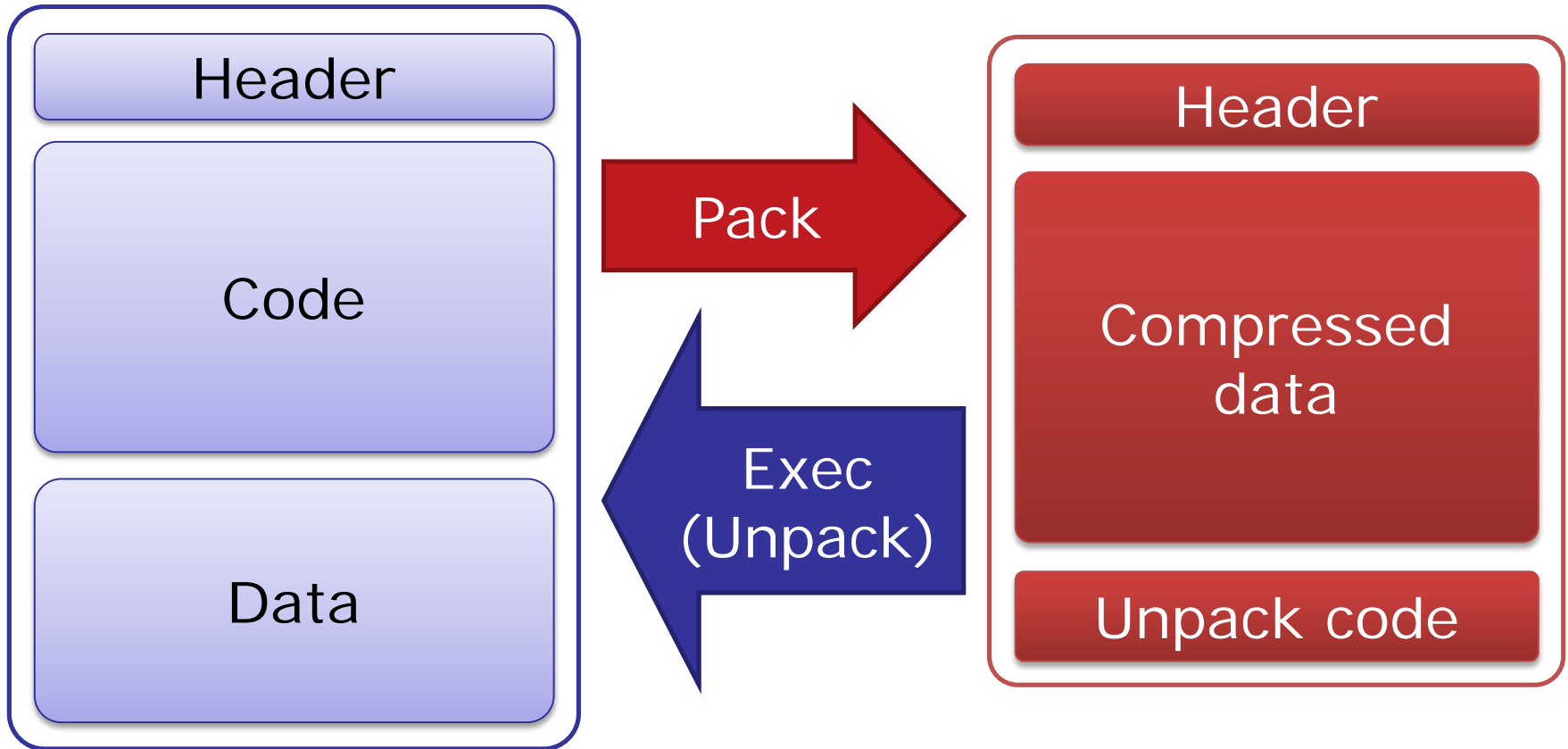
NAKATSURU You

Windows Malware Analysis

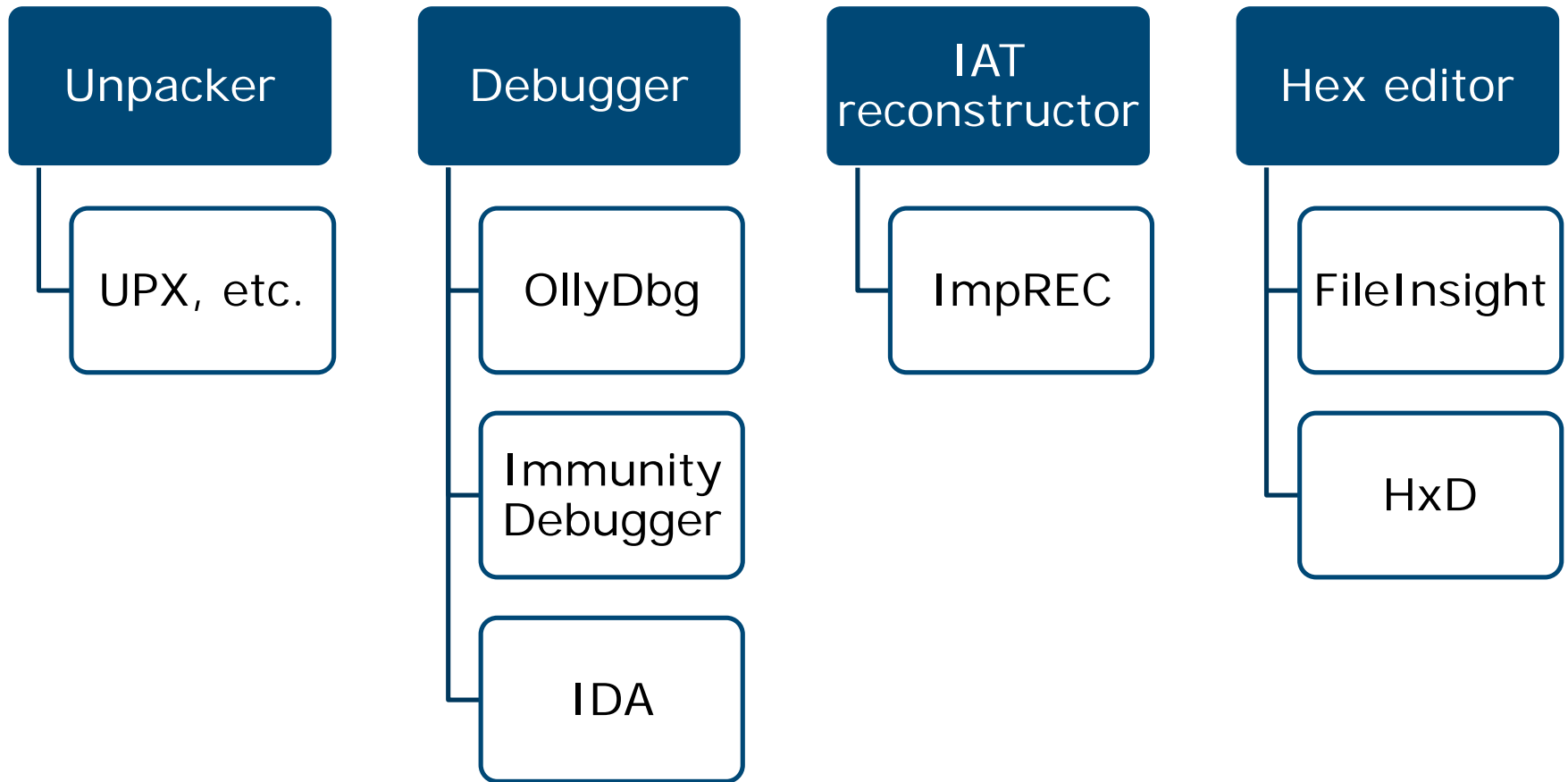


What Packing/Unpacking is

- "Pack" original code for compression/obfuscation

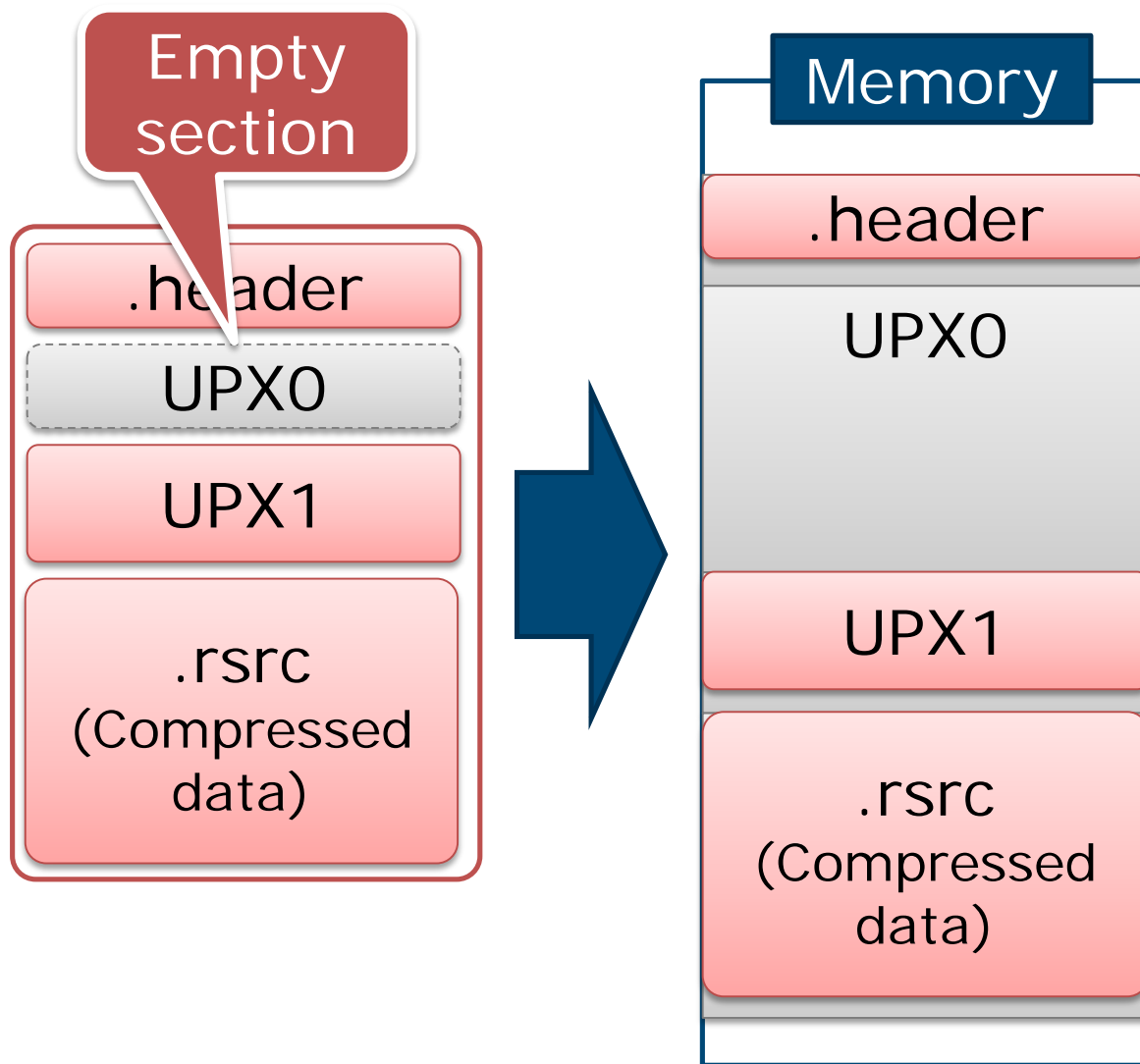


Unpacking Tools

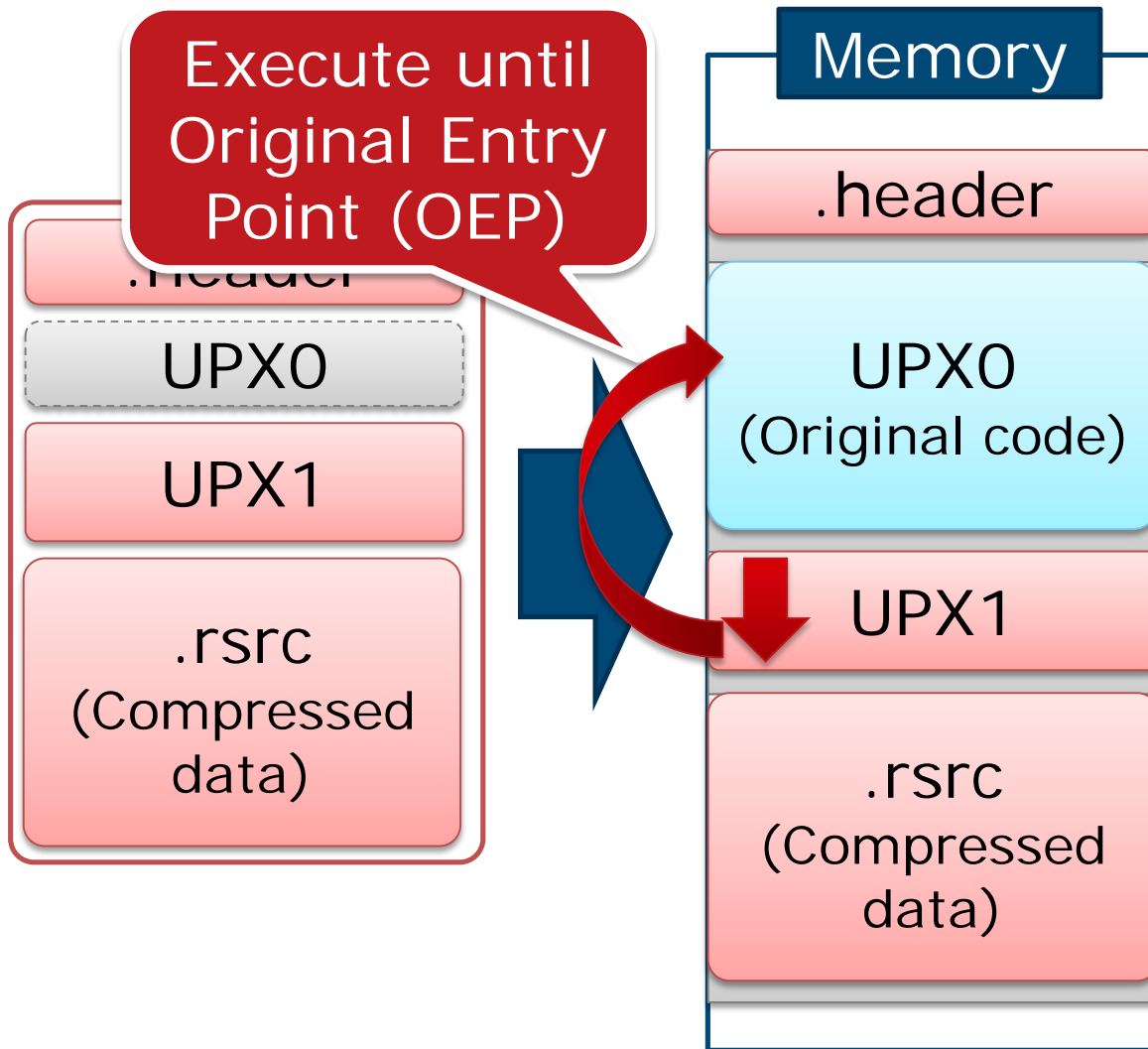


CLASSIC UNPACKING

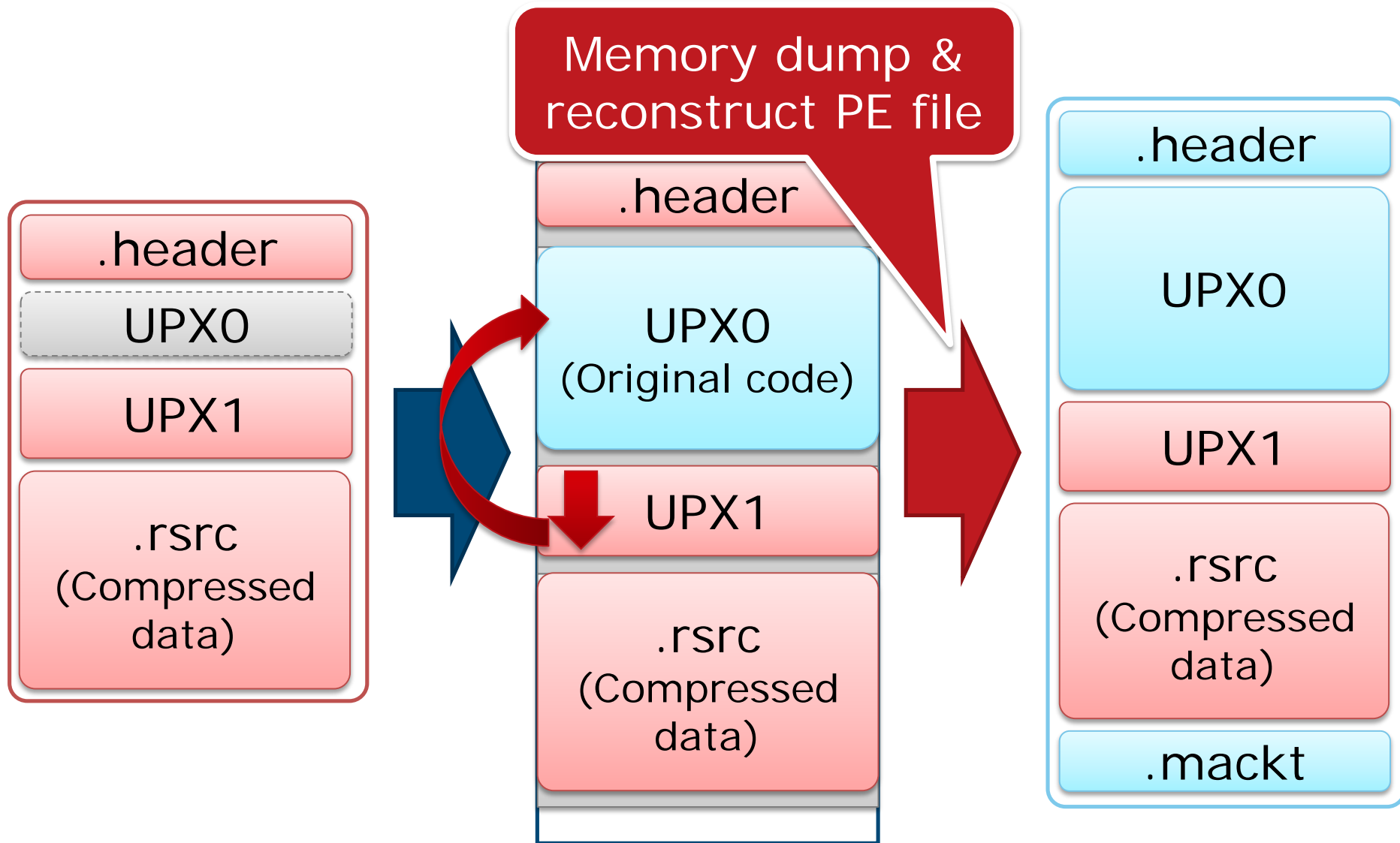
What "Classic Unpacking" is



What "Classic Unpacking" is



What "Classic Unpacking" is



Classic Unpacking Flow

1. Execute unpack code

- Find OEP

2. Dump as a PE file

- reconstruct PE header, etc.

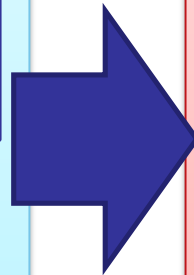
3. Reconstruct Import Address Table (IAT)

Reconstructing IAT

PE File

kernel32.dll
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess

Import
Directory



IAT on Memory

extrn GetProcAddress: dword
extrn VirtualProtect: dword
extrn VirtualAlloc: dword
extrn VirtualFree: dword
extrn ExitProcess: dword

IAT

Reconstructing IAT

PE File

kernel32.dll
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess

IAT

IAT on Memory

```
extrn GetProcAddress:dword  
extrn VirtualProtect  
extrn VirtualAlloc:dword  
extrn VirtualFree:dword  
extrn ExitProcess:dword
```

Created by
unpack code

```
extrn RegQueryValueExA:dword  
extrn RegSetValueExA:dword  
extrn RegEnumKeyA:dword  
extrn RegEnumValueA:dword  
extrn RegOpenKeyExA:dword  
extrn RegDeleteKeyA:dword  
extrn RegDeleteValueA:dword  
extrn RegCloseKey:dword  
extrn RegCreateKeyExA:dword
```

Reconstructing IAT

PE File

kernel32.dll
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess

Can not import
required APIs

IAT

IAT on Memory

extrn GetProcAddress:dword
extrn VirtualProtect:dword
extrn VirtualAlloc:dword
extrn VirtualFree:dword
extrn ExitProcess:dword

extrn RegQueryValueExA:dword
extrn RegSetValueExA:dword
extrn RegEnumKeyA:dword
extrn RegEnumValueA:dword
extrn RegOpenKeyExA:dword
extrn RegDeleteKeyA:dword
extrn RegDeleteValueA:dword
extrn RegCloseKey:dword
extrn RegCreateKeyExA:dword

Reconstructing IAT

PE File

kernel32.dll
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess

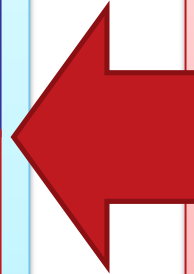
RegQueryValueExA
RegSetValueExA
RegEnumKeyA
RegEnumValueA
RegOpenKeyExA
RegDeleteKeyA
RegDeleteValueA
RegCloseKey
RegCreateKeyExA

IAT

IAT on Memory

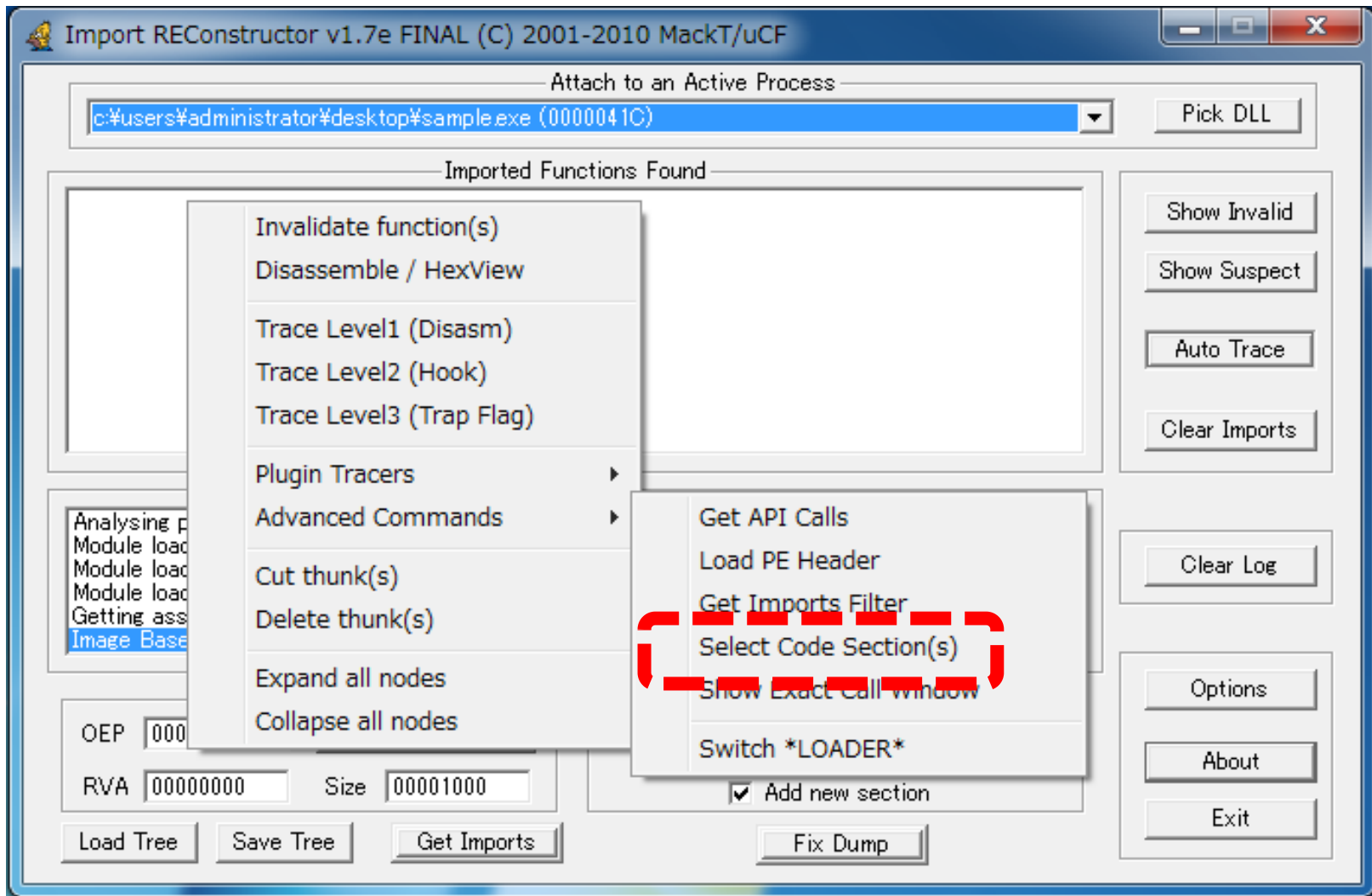
extrn **GetProcAddress**:dword
extrn **VirtualProtect**:dword
extrn **VirtualAlloc**:dword
extrn **VirtualFree**:dword
extrn **ExitProcess**:dword

extrn **RegQueryValueExA**:dword
extrn **RegSetValueExA**:dword
extrn **RegEnumKeyA**:dword
extrn **RegEnumValueA**:dword
extrn **RegOpenKeyExA**:dword
extrn **RegDeleteKeyA**:dword
extrn **RegDeleteValueA**:dword
extrn **RegCloseKey**:dword
extrn **RegCreateKeyExA**:dword



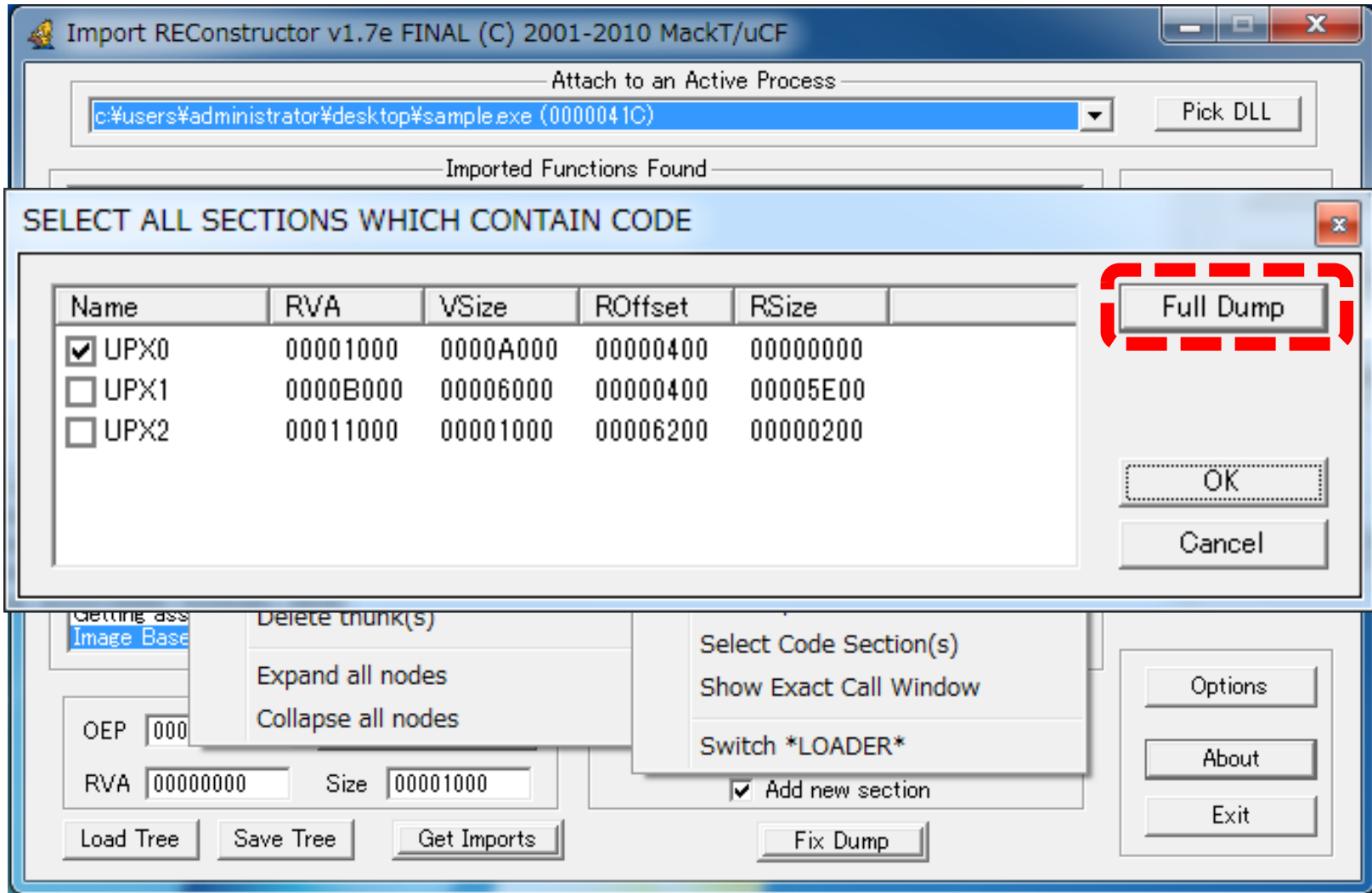
Classic Unpacking

■ Reconstruct PE file



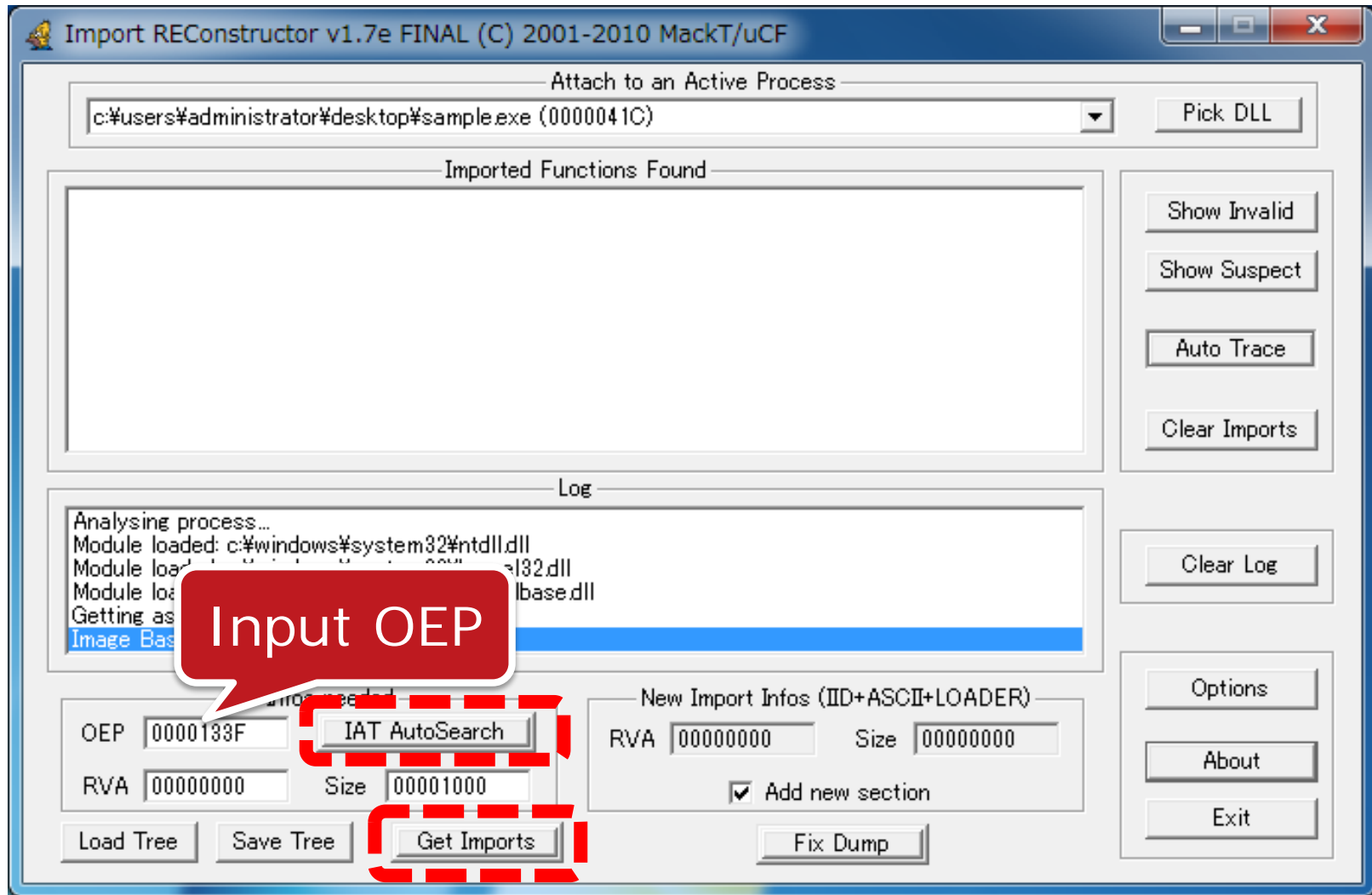
Classic Unpacking

■ Reconstruct PE file



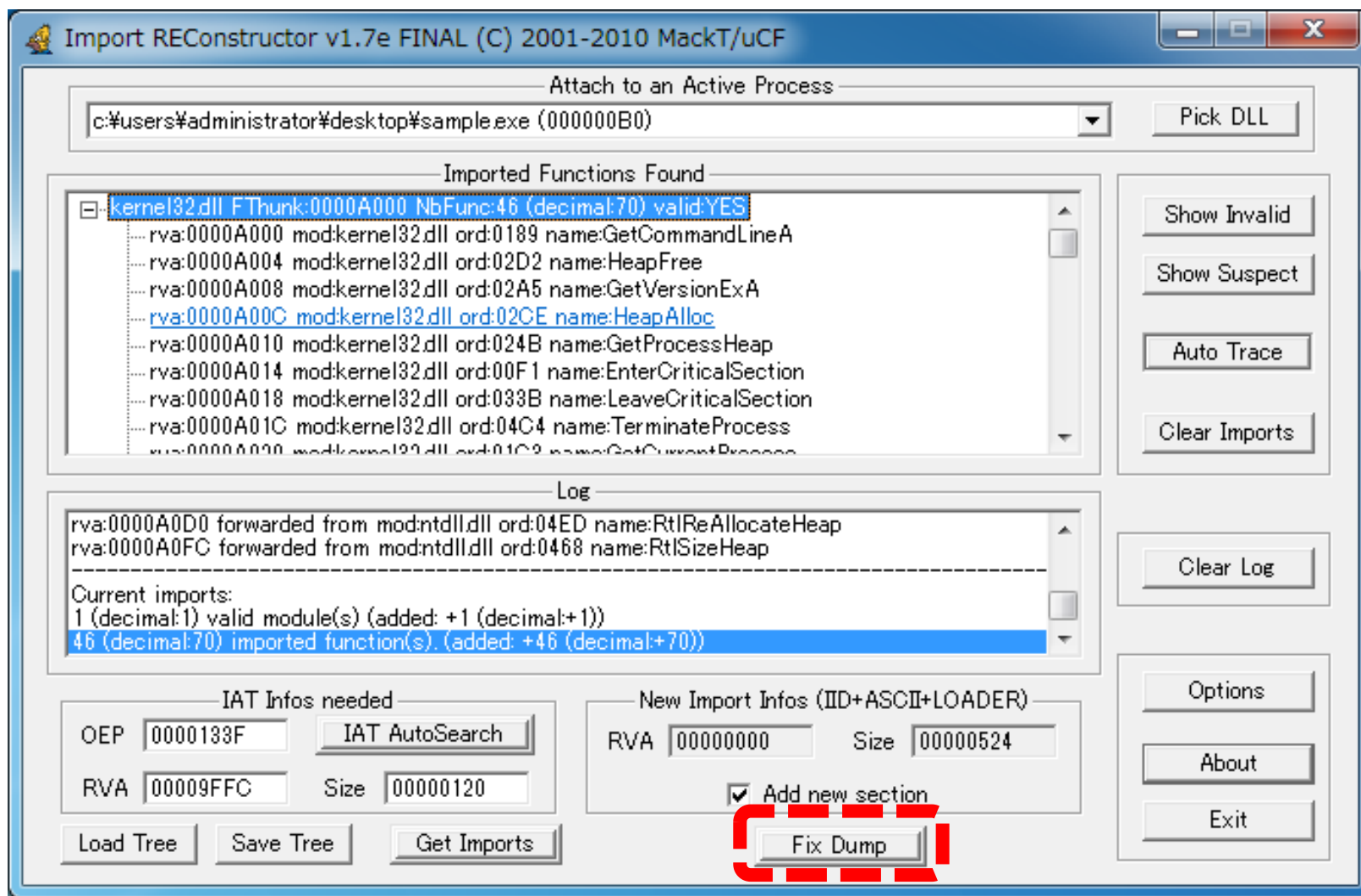
Classic Unpacking

■ Reconstruct IAT

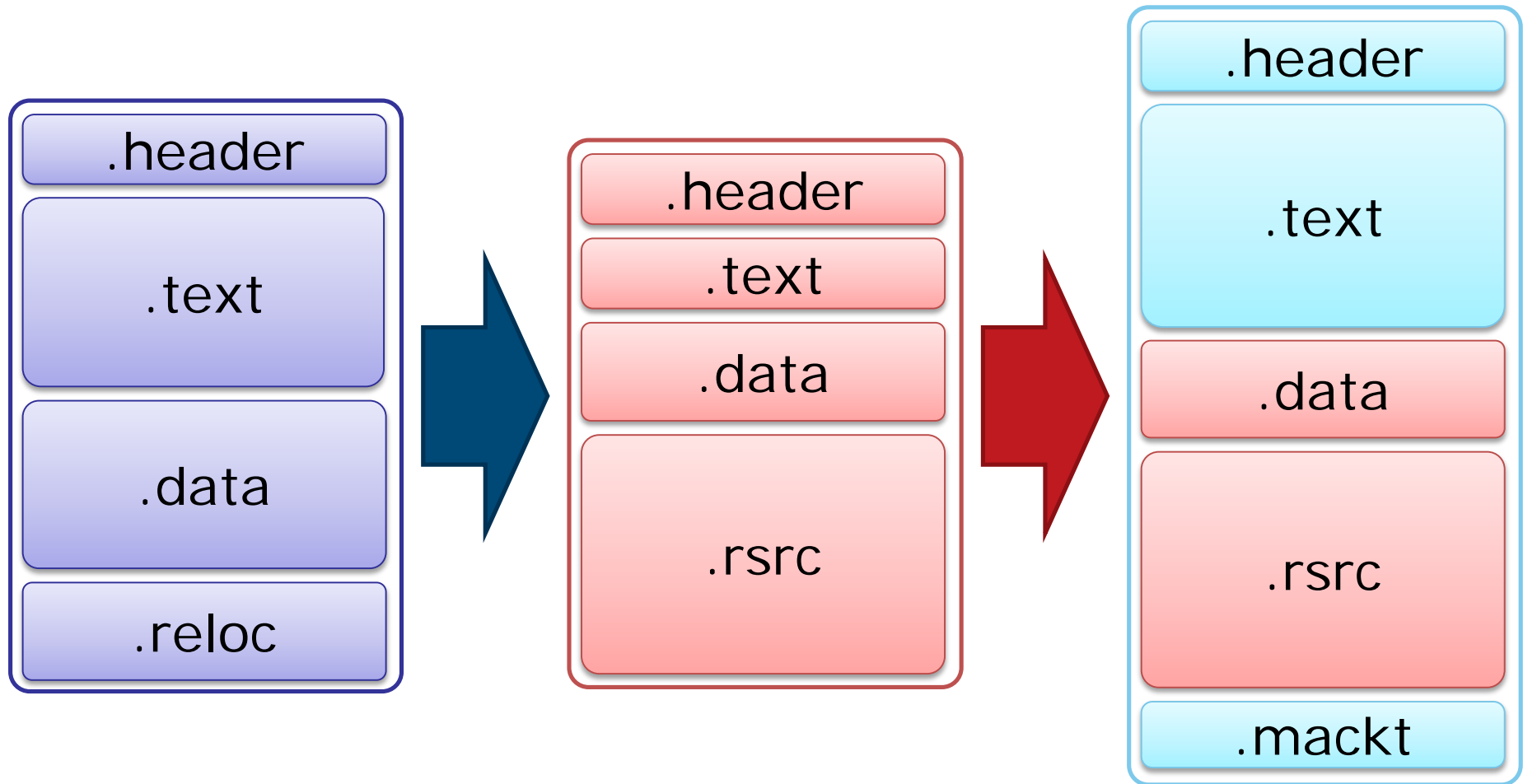


Classic Unpacking

■ Reconstruct IAT

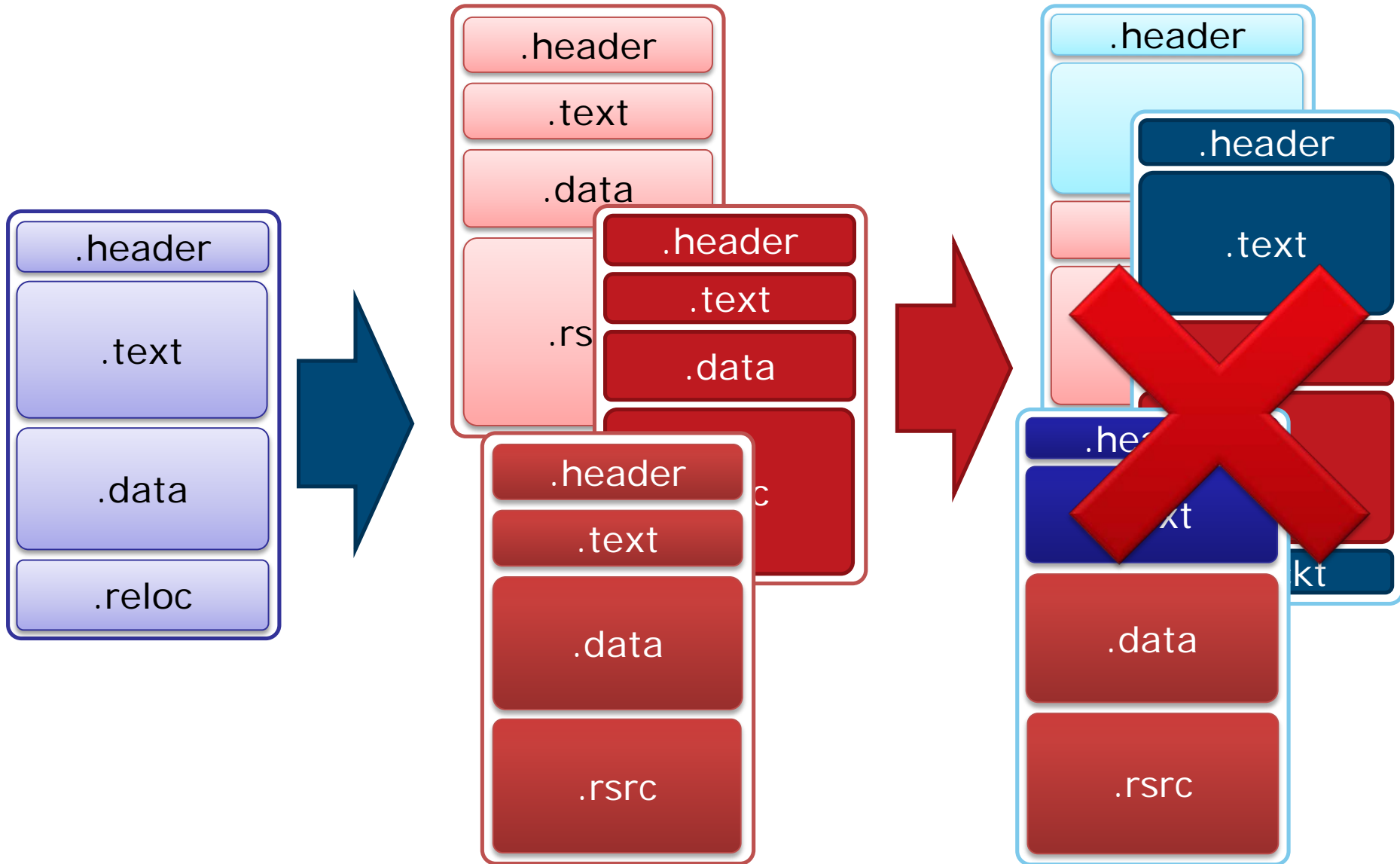


Classic Unpacking Issue



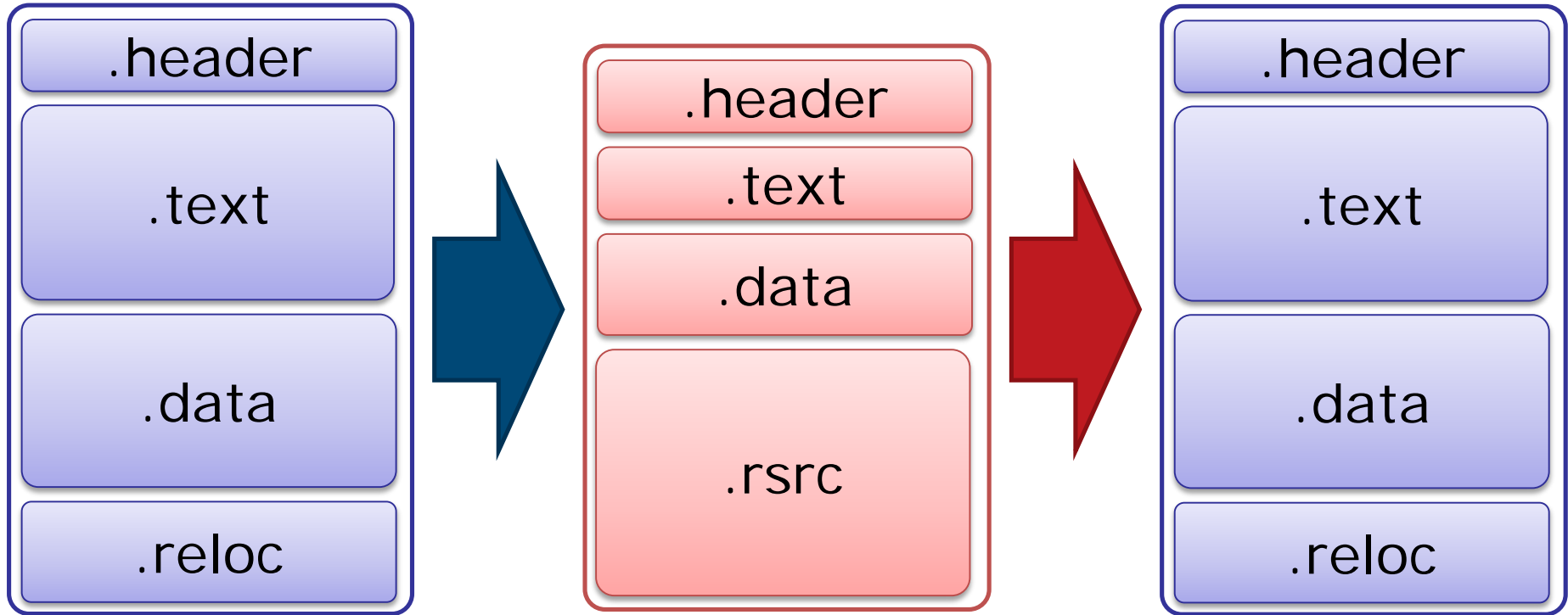
hash("Original") \neq hash("Unpacked")

Classic Unpacking Issue



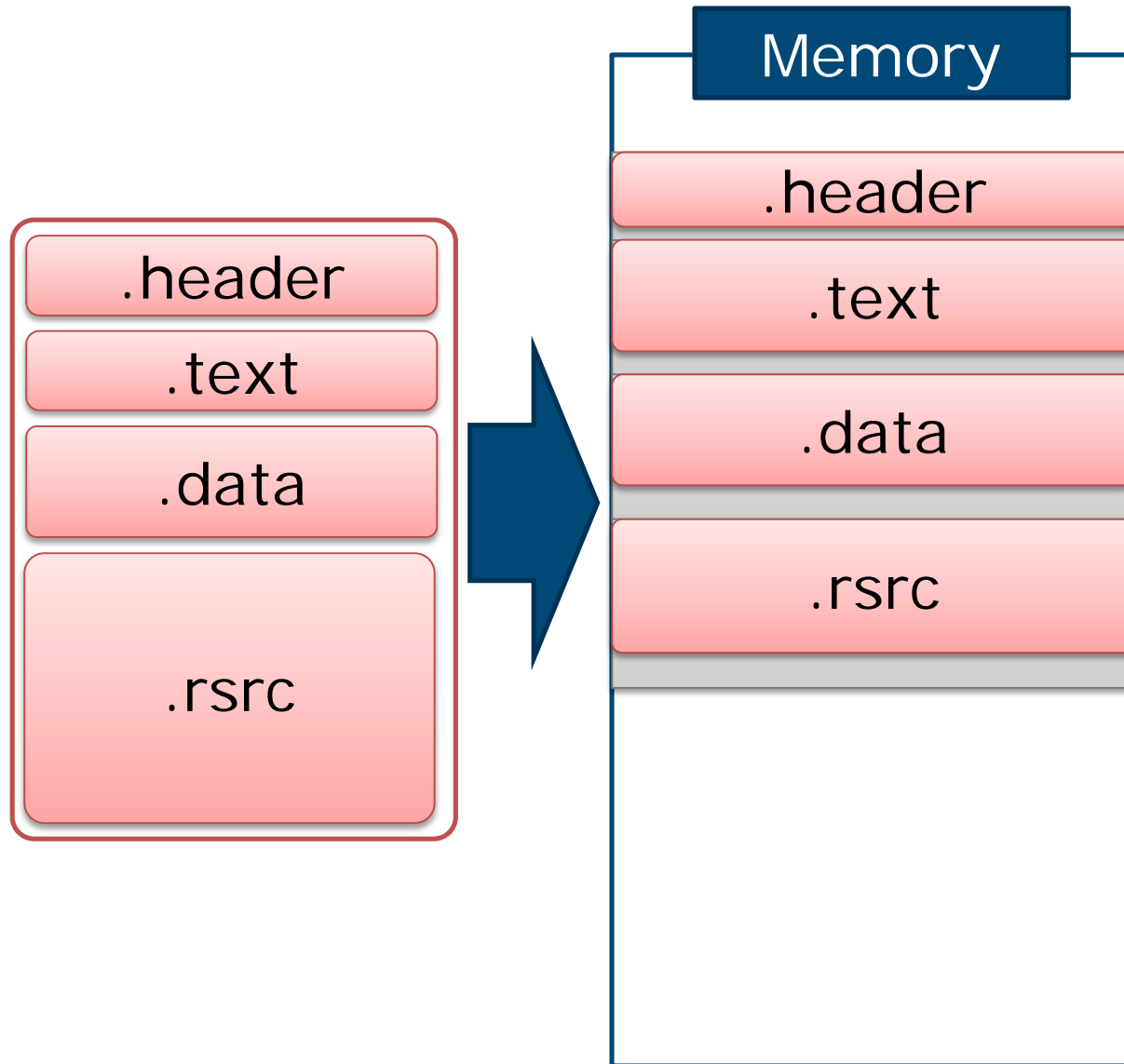
PERFECT UNPACKING

Concept

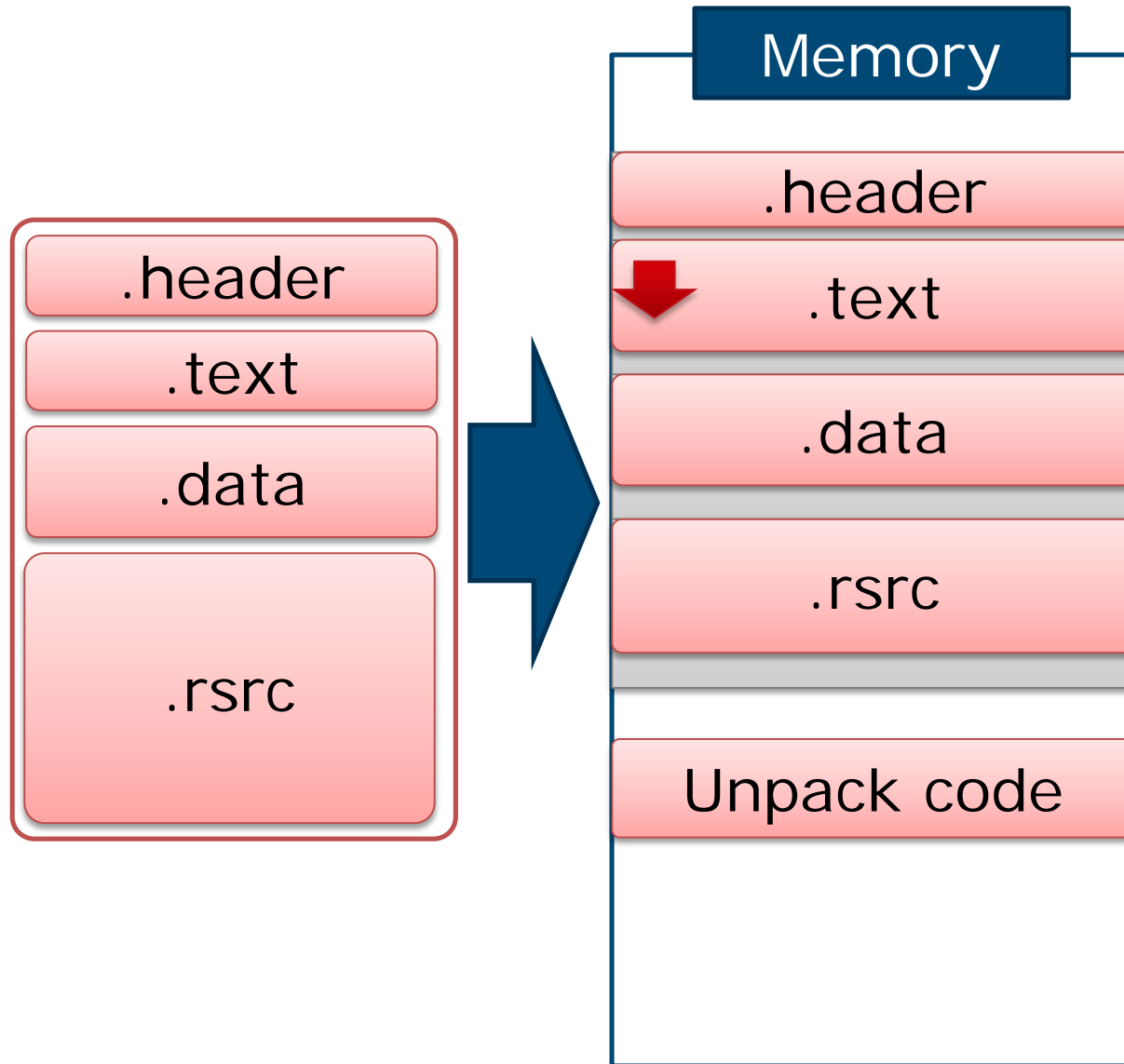


hash("Original") == hash("Unpacked")

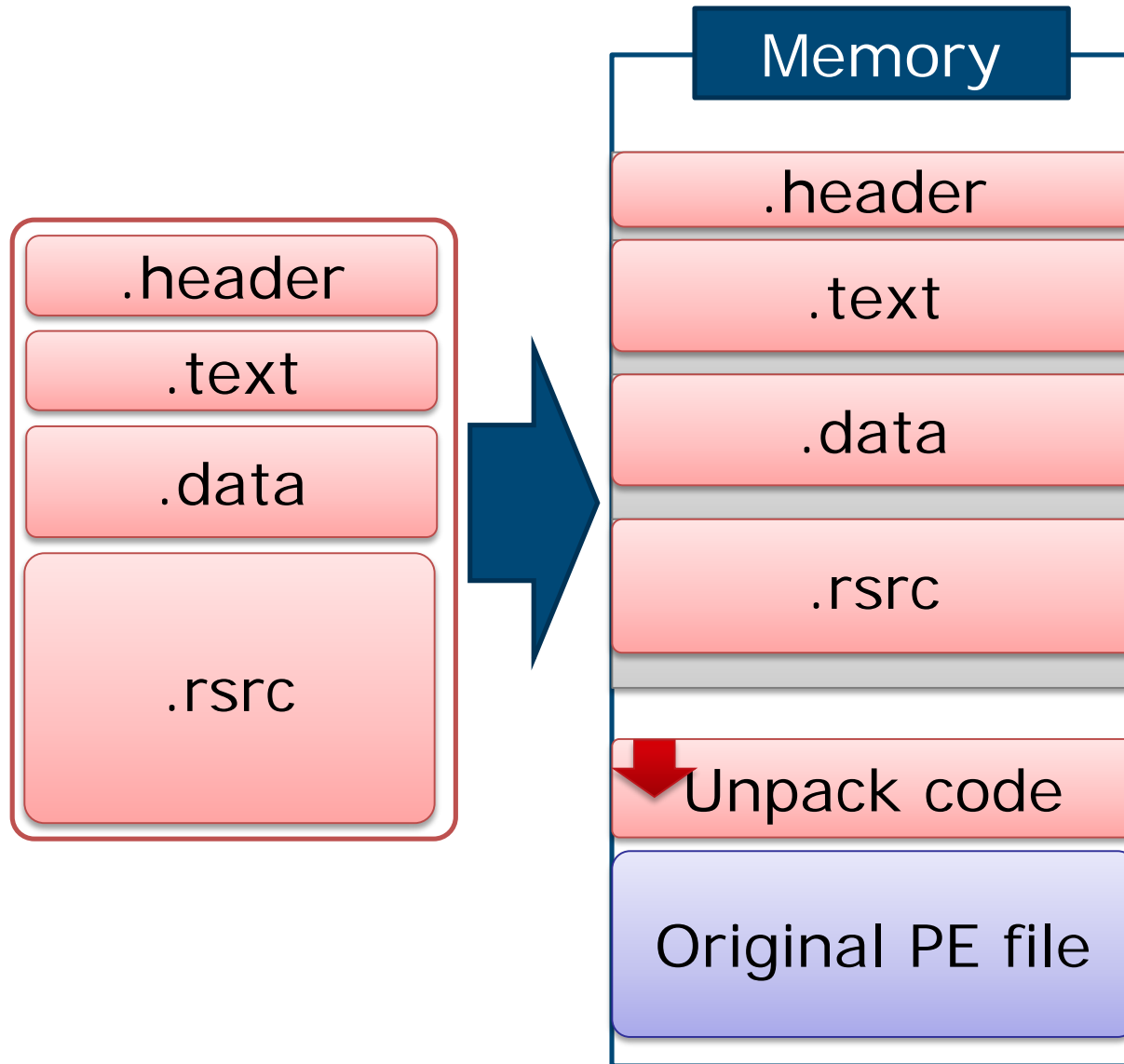
Recent Packer



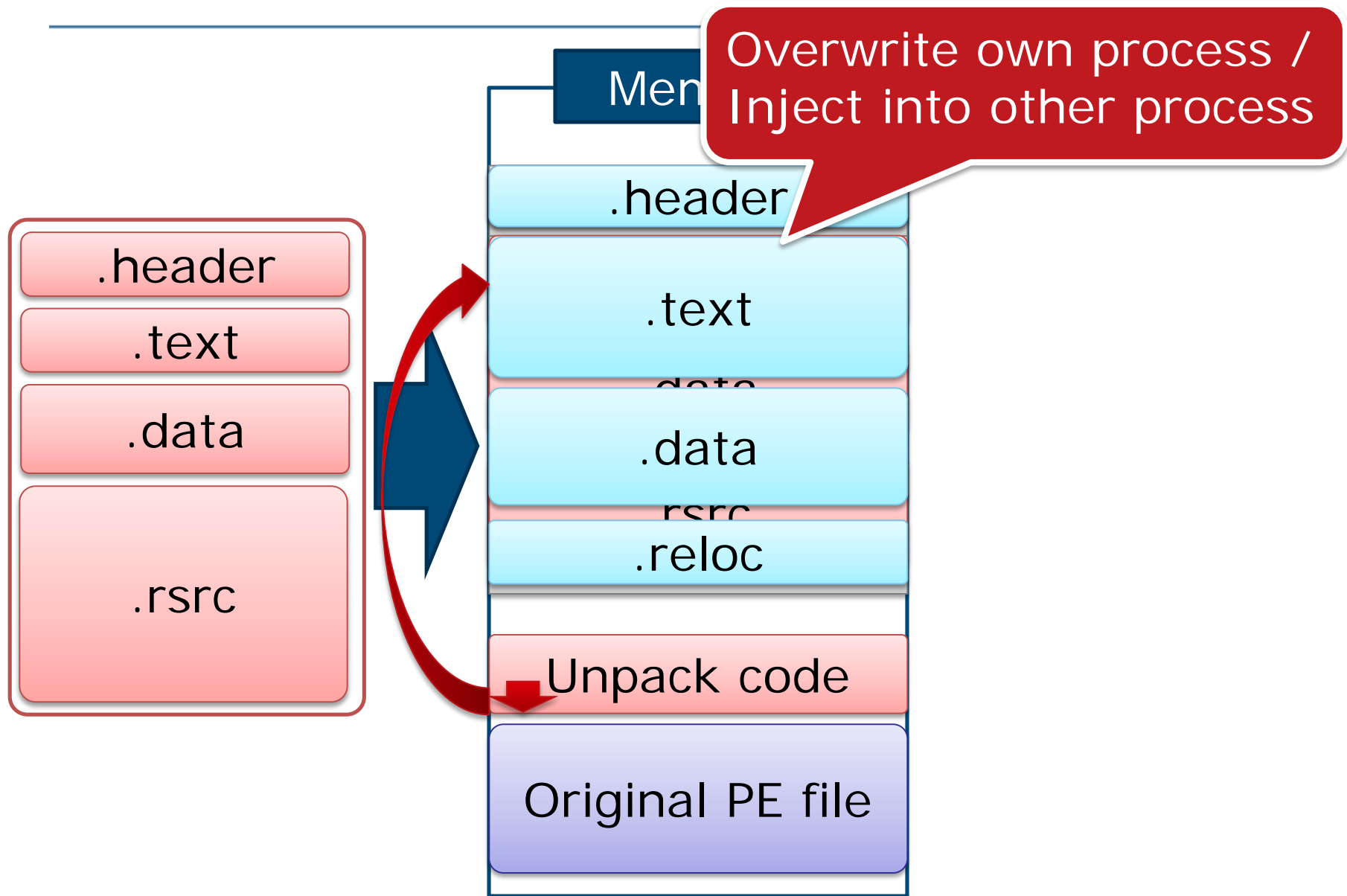
Recent Packer



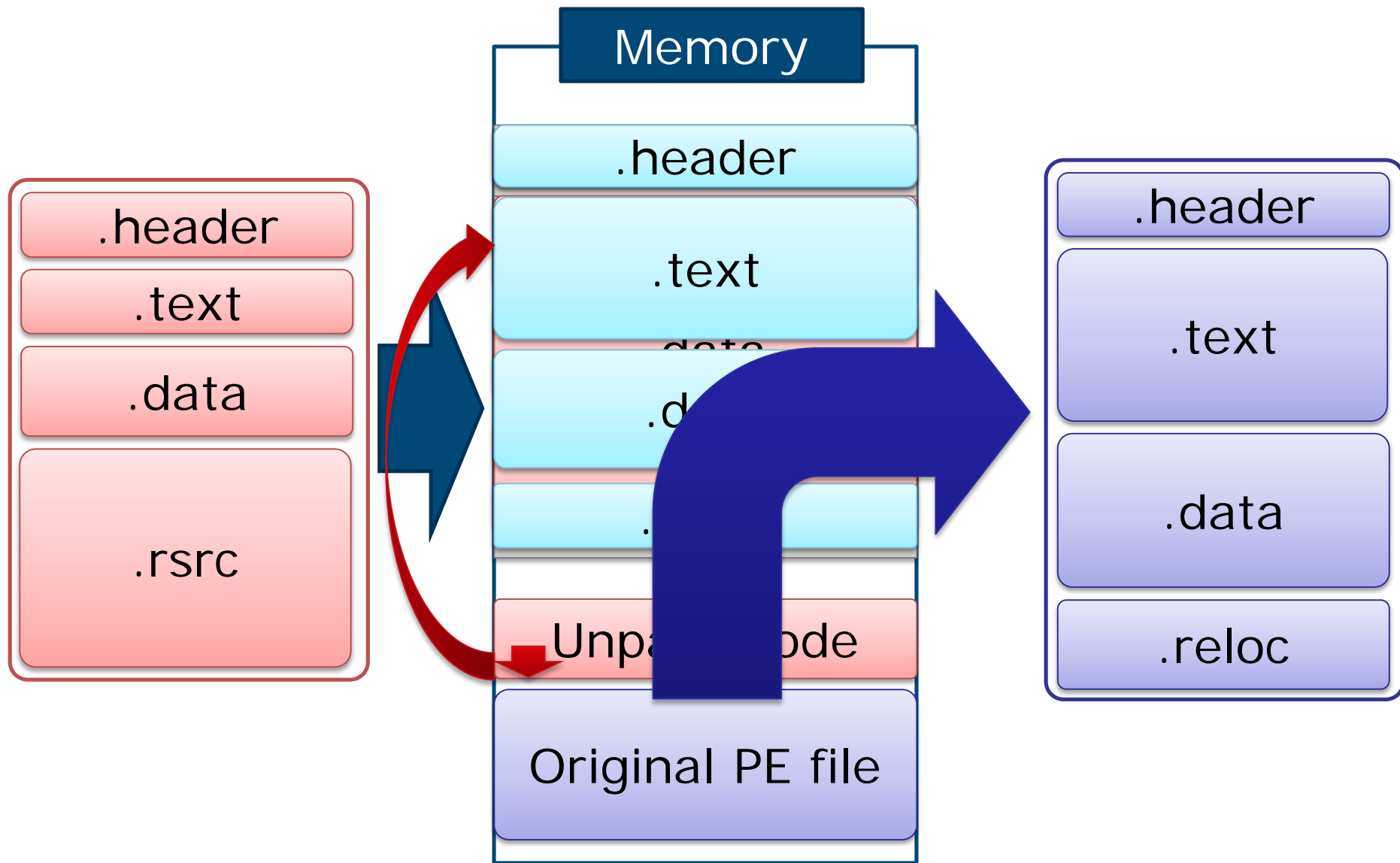
Recent Packer



Recent Packer



Recent Packer



Perfect Unpacking Flow

1. Execute unpack code

- Let unpack code unpack original PE file



2. Dump memory section contains original PE file



3. Trim PE file

1. Unpack Code Execution

- Set breakpoints on

Windows APIs

- WriteProcessMemory
- ZwWriteVirtualMemory
- CreateProcessW
- VirtualFree / RtlFreeHeap
- etc.

PE header

- Hardware breakpoint on "M"

2. Dumping Memory Section

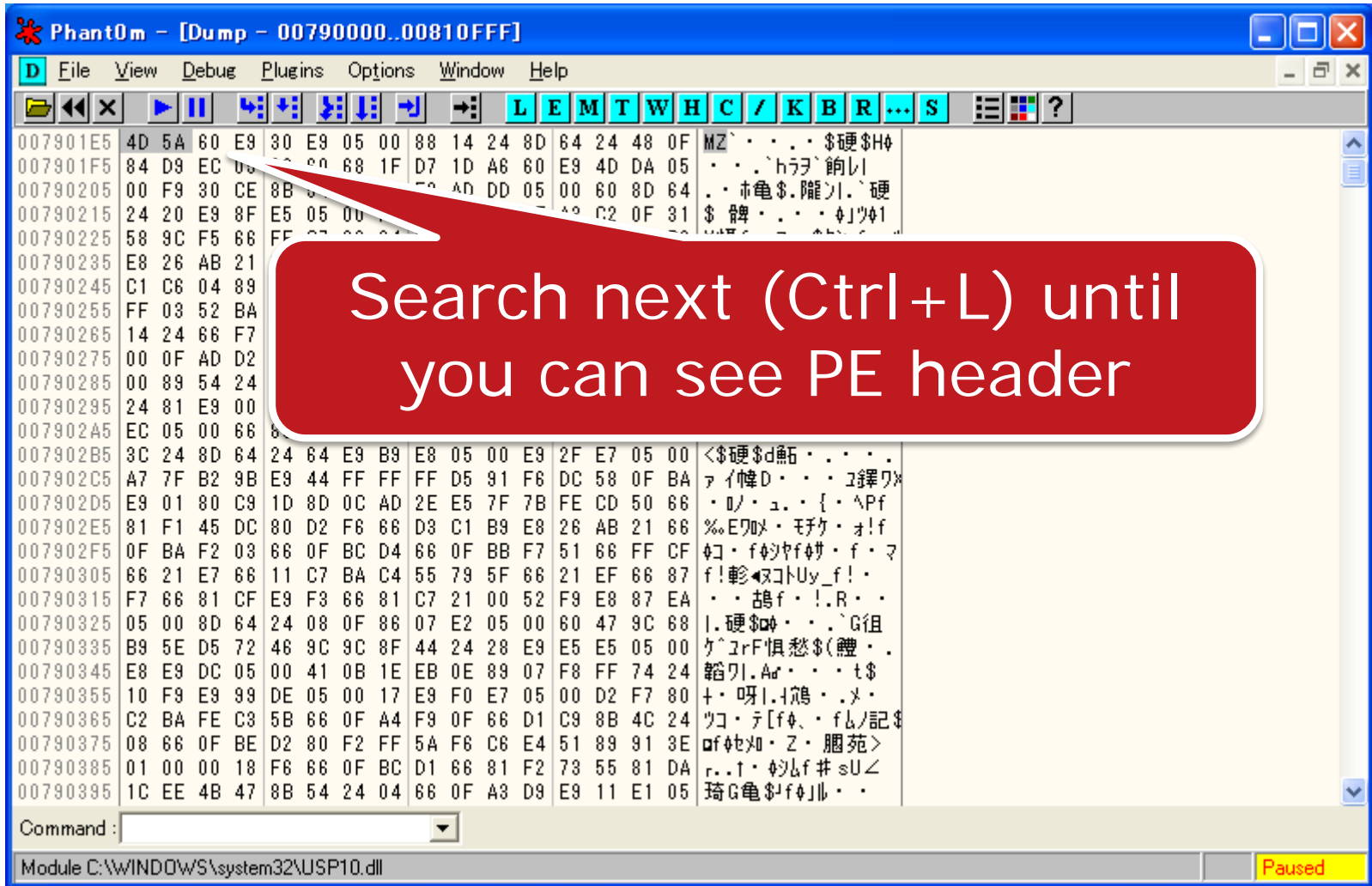
■ Search "MZ" string

The screenshot shows the Phantom memory map tool interface. A dialog box titled "Enter binary string to search for" is open, allowing the user to search for a specific binary string. The dialog has three input fields: ASCII (containing "MZ"), UNICODE (empty), and HEX +02 (containing "4D 5A"). There are also checkboxes for "Entire block" and "Case sensitive", both of which are checked. The "OK" and "Cancel" buttons are at the bottom of the dialog. A red callout bubble with the text "Ctrl + B" points to the dialog box. The background shows a memory map table with columns for Address, Size, Owner, Section, Contains, Type, and Access. The current module is identified as "Module C:\WINDOWS\system32\USP10.dll" and the tool is in a "Paused" state.

Address	Size	Owner	Section	Contains	Type	Acc
00010000	00001000				Priv	RW
00020000	00001000				Priv	RW
0006C000	00001000				Priv	RW
0006D000	00003000					
00070000	00003000					
00080000	00001000					
00090000	00005000					
000A0000	00006000					
000B0000	00003000					
000C0000	00016000					
000E0000	00041000					
00130000	00041000					
00180000	00006000					
00190000	00041000					
001E0000	00004000					
002A0000	00002000					
002B0000	00103000					
003C0000	00008000					
003D0000	00001000				Priv	RW
003E0000	00001000				Priv	RW
00400000	00001000	ufugi			Imag	RWE
00401000	0005F000	ufugi	.blfm	code	Imag	RWE
00460000	00002000	ufugi	.yhlj	data	Imag	RWE
00462000	00010000	ufugi	.mbus	imports	Imag	RWE
00472000	00010000	ufugi	.qjlx	resources	Imag	RWE
00490000	0007C000				Map	R E
00790000	00081000				Priv	RWE

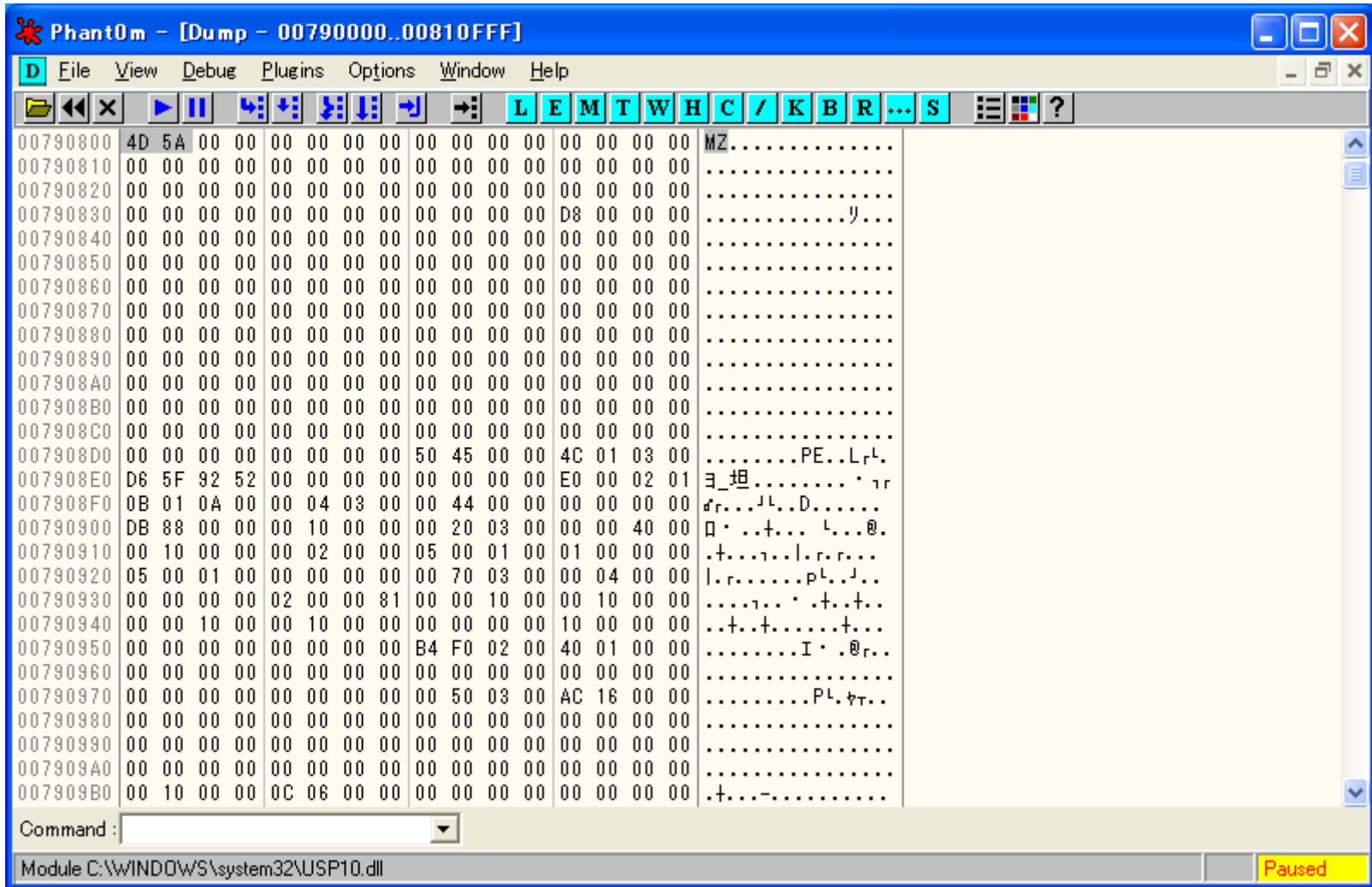
2. Dumping Memory Section

■ Search "MZ" string



2. Dumping Memory Section

■ Search "MZ" string



2. Dumping Memory Section

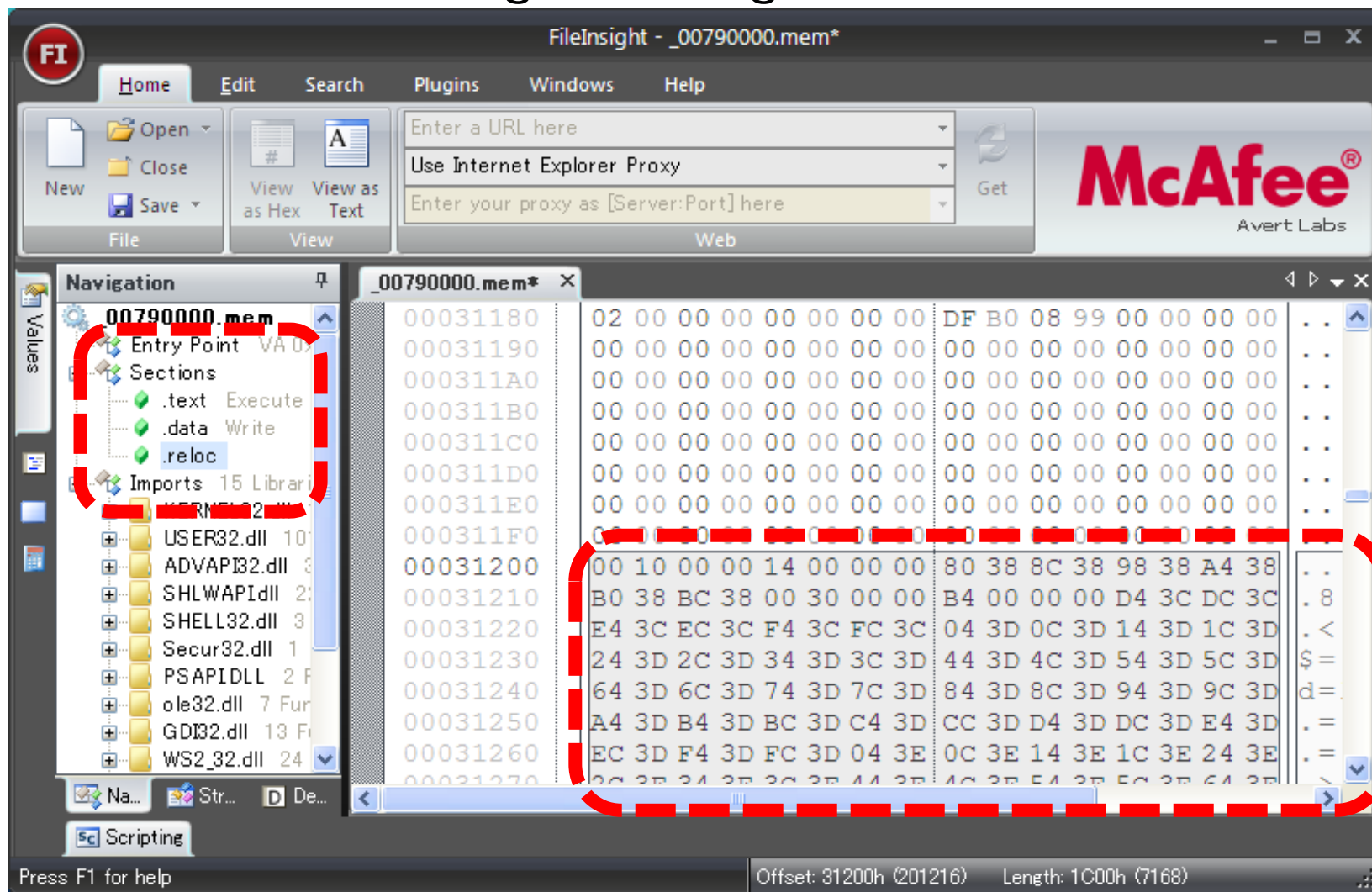
■ Search "MZ" string

The screenshot shows the PhantOm memory dump interface. The title bar reads "PhantOm - [Dump - 00790000..00810FFF]". The menu bar includes "File", "View", "Debug", "Plugins", "Options", "Window", and "Help". The toolbar contains various navigation and search icons, with "L E M T W H C / K B R ... S" highlighted. The main display area shows a memory dump with addresses on the left, hex bytes in the middle, and ASCII characters on the right. A red callout box with the text "Raw address" points to the address \$+3E0. A red dashed box highlights the memory location starting at \$+400, where the hex bytes "5A 5A" (representing "MZ") are visible. The status bar at the bottom shows "Module C:\WINDOWS\system32\USP10.dll" and "Paused".

Address	Hex	ASCII
\$+300	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+310	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+320	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+330	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+340	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+350	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+360	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+370	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3A0	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3B0	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3C0	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+3F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
\$+400	5A 5A 03 00 34 08 03 00 44 08 03 00 5E 08 03 00	"MZ....."
\$+410	7E 08 03 00 9A 08 03 00 B6 08 03 00 CA 08 03 00	~MZ.....
\$+420	DE 08 03 00 F4 08 03 00 0C 09 03 00 1E 09 03 00	~MZ.....
\$+430	38 09 03 00 4E 09 03 00 60 09 03 00 76 09 03 00	8.MZ.....
\$+440	8A 09 03 00 A2 09 03 00 BA 09 03 00 D2 09 03 00	.MZ.....
\$+450	E4 09 03 00 1C 0A 03 00 2C 0A 03 00 48 0A 03 00	.MZ.....
\$+460	64 0A 03 00 78 0A 03 00 90 0A 03 00 9E 0A 03 00	d.MZ.....
\$+470	B0 0A 03 00 C0 0A 03 00 D0 0A 03 00 00 00 00 00	~MZ.....
\$+480	9A 0F 03 00 82 0F 03 00 60 0F 03 00 42 0F 03 00	.MZ.....
\$+490	2C 0F 03 00 1A 0F 03 00 02 0F 03 00 E0 0E 03 00	.MZ.....
\$+4A0	00 00 00 00 9E 0D 03 00 B4 0D 03 00 CA 0D 03 00MZ.....
\$+4B0	D8 0D 03 00 E8 0D 03 00 F8 0D 03 00 04 0E 03 00	..MZ.....

3. Trimming PE file

■ Parse PE file using FileInsight



Consideration

0. Limited availability

- Depends on packer's implementation

1. Unpacking code execution

- Debugger & VM detection
- **Breakpoint detection**

3. Trimming PE file

- Overlay data
 - Data used by malware
 - e.g. ZeuS variants
- **Digital signature**

Demo Movie

- Get same original file from different binaries using "Perfect Unpacking"

CONCLUSION

Summary

Classic unpacking issue

- Unpacked file's hash value depends on analyst/tools

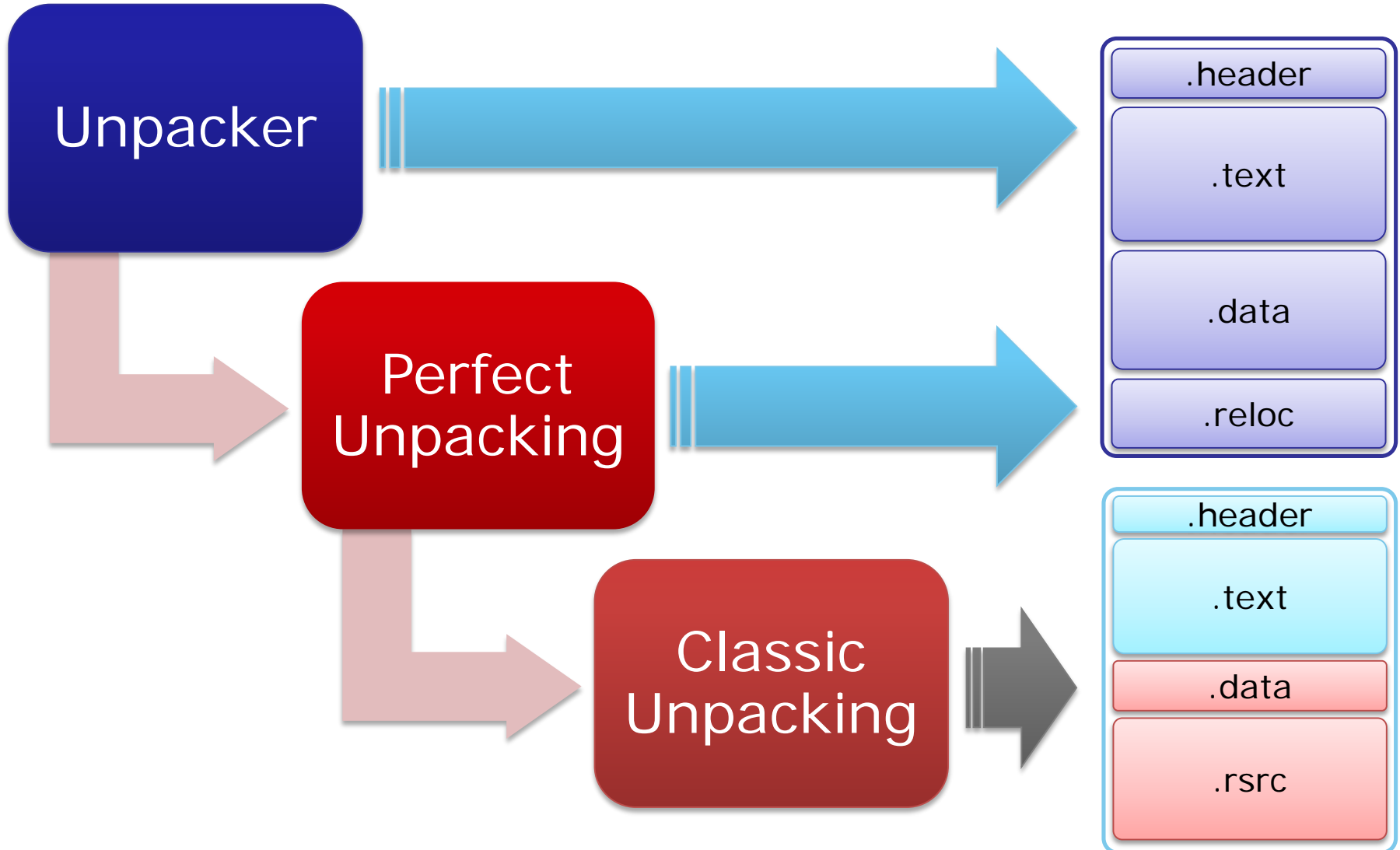
Resent packer implementation

- Packed malware contains original PE file

We have to perform "Perfect Unpacking"

- Dump original PE file from virtual memory

Recommended Unpacking Flow



Thank you!

Contact

- aa-info@jpcert.or.jp
- <https://www.jpcert.or.jp>

Incident report

- info@jpcert.or.jp
- <https://www.jpcert.or.jp/form/>