Binary art
Byte-ing the PE that *fails* you

Ange Albertini
http://corkami.com

3rd November 2012
Lucerne, Switzerland

extended edition

- the **presentation** deck had 60+ slides
- this **one** has 140+
  - many extra explanation slides
  - many extra examples

agenda


what's a PE?

the problem, and my approach


overview of the PE format


classic tricks


new tricks

Portable ***E***xecutable

based on

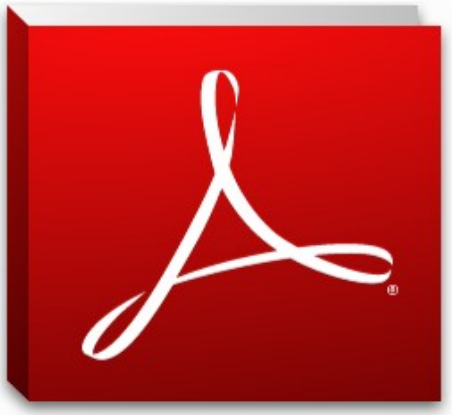
Common ***O***bject ***F***ile ***F***ormat

Windows executables and more

- since 1993, used in *almost* every executables
  - 32bits, 64bits, .Net
  - DLL, drivers, ActiveX...
- also used as data container
  - icons, strings, dialogs, bitmaps...

omnipresent in Windows

also EFI boot, CE phones, Xbox,...
(but not covered here)



AcroRd32

calc

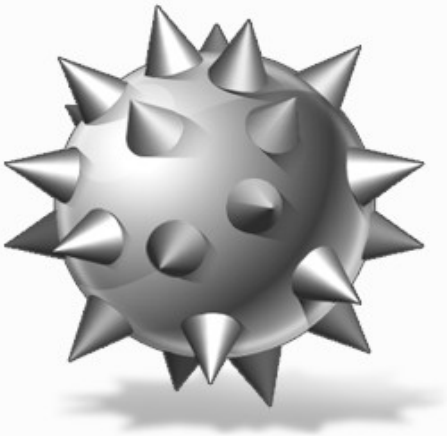
chrome

cmd

explorer

firefox

iexplore

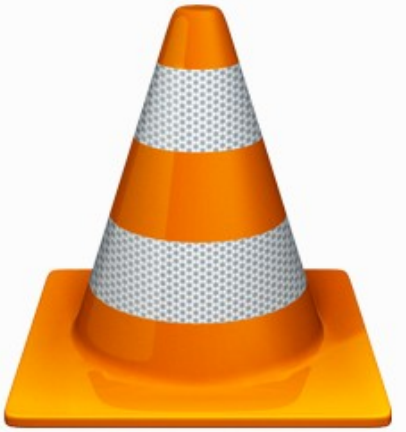


MineSweeper

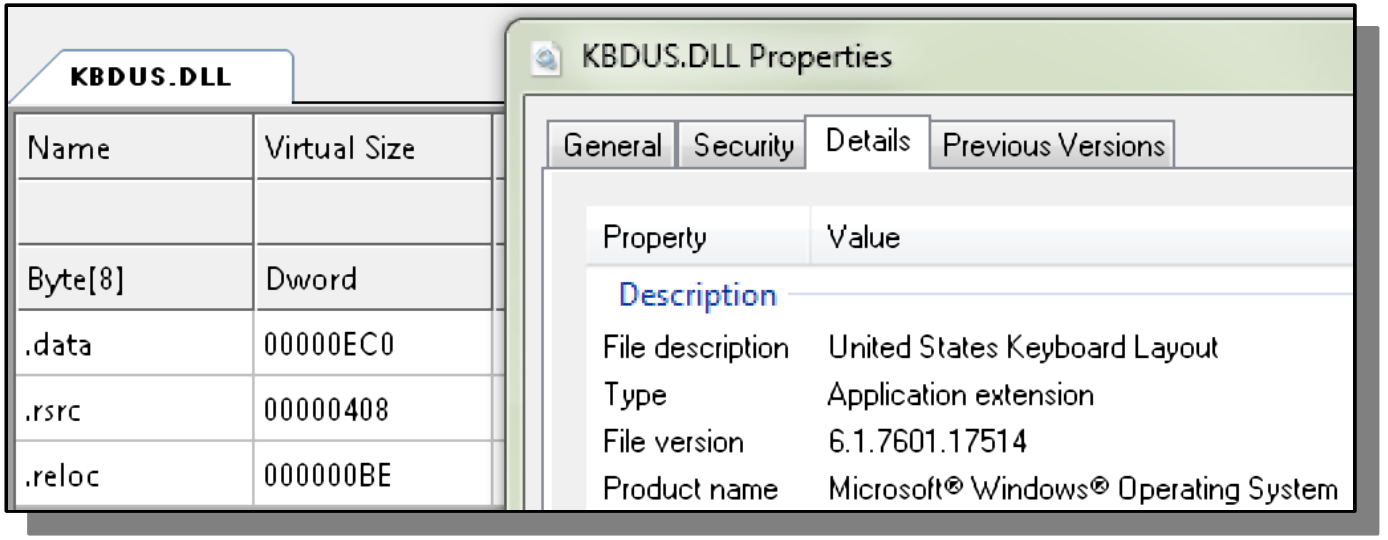
mspaint

notepad

Solitaire



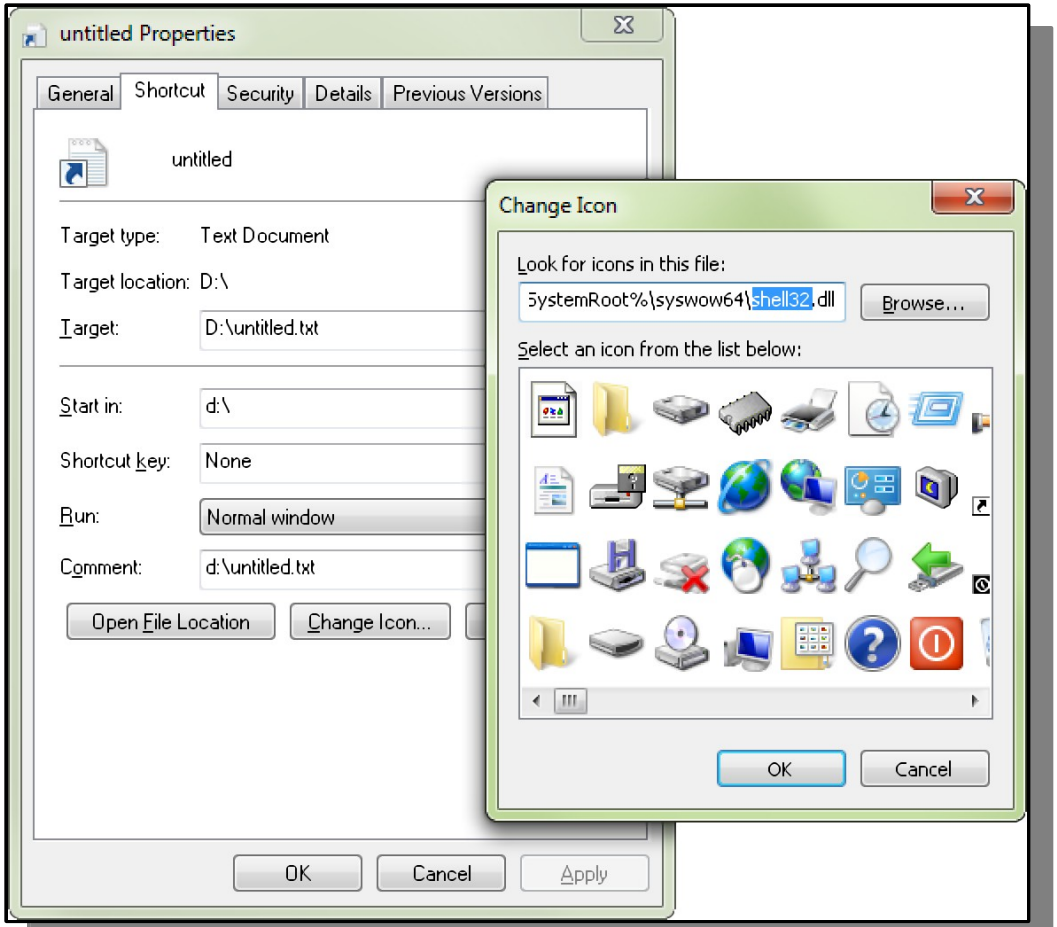
vlc



Sections

name	va	vsize	raw size	flags
.rsrc	0x1000	0x6000	0x5400	R-- IDATA

id	lang	string
300	1033	Access letters, reports, notes, and other kinds of documents.
301	1033	Displays recently opened documents and folders.
302	1033	Play music and other audio files.
303	1033	View and organize digital pictures.
304	1033	See the disk drives and other hardware connected to your computer.
305	1033	Access libraries and folders shared by other people in your homegroup.
307	1033	Change settings and customize the functionality of your computer.
312	1033	Play and manage games on your computer.
316	1033	Choose default programs for web browsing, e-mail, playing music, and other activities.
317	1033	Open your personal folder.
318	1033	See the available wireless networks, dial-up, and VPN connections that you can connect
319	1033	See installed printers and add new ones.
320	1033	Pinned
321	1033	Frequently used programs
322	1033	Use tab to move between sections, including the first and second panes and the power b
323	1033	Pin %s
324	1033	Unpin %s
325	1033	Launch Windows Security Options to Change Password, Switch User, or Start Task Manager
326	1033	Pinned
327	1033	Recent
328	1033	Frequent
329	1033	Tasks
330	1033	&Unpin from this list
331	1033	P&in to this list
332	1033	Unpin from Start menu
333	1033	Pin to Start menu
334	1033	Show recent or related items for %s
335	1033	Show recent or related items.
336	1033	Hide recent or related items for %s
337	1033	Hide recent or related items.
338	1033	Pin this program to taskbar
339	1033	Unpin this program from taskbar
340	1033	Location: %s (%s)
341	1033	Open



PE

universal

Windows binary

since 1993

PE101 a windows executable walkthrough

Ange Albertini
corkami.com

Dissected PE

Loading process

❶ Headers

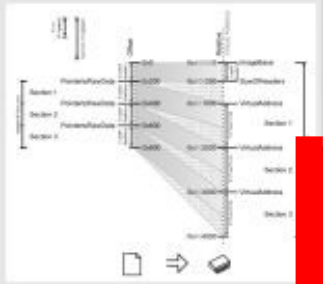
the *DOS Header* is parsed
the PE Header is parsed
(its offset is DOS Header's e_lfanew)
the Optional Header is parsed
(it follows the PE Header)

❷ Sections table

Sections table is parsed
(it is located at offset (*OptionalHeader*) + *SizeOfOptionalHeader*)
it contains *NumberOfSections* elements
it is checked for validity with alignments:
FileAlignments and *SectionAlignments*

❸ Mapping

the file is mapped in memory according to:
the *ImageBase*
the *SizeOfHeaders*
the Sections table

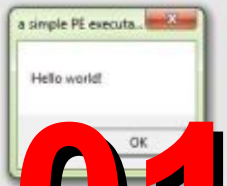


❹ Imports

DataDirectories are parsed
they follow the OptionalHeader
their number is *NumOfRVAAndSizes*
imports are always #2
imports are parsed
each descriptor specifies a *DLLname*
this DLL is loaded in memory
IAT and *INT* are parsed simultaneously
for each API in *INT*
its address is written in the IAT entry

❺ Execution

Code is called at the *EntryPoint*
the calls of the code go via the IAT to the APIs



Notes

MZ HEADER aka DOS_HEADER
Starts with 'MZ' (initials of *Mark Zbikowski* MS-DOS developer)

PE HEADER aka IMAGE_FILE_HEADERS / COFF file header
Starts with 'PE' (Portable Executable)

OPTIONAL HEADER aka IMAGE_OPTIONAL_HEADER
Optional only for non-standard PEs but required for executables

RVA Relative Virtual Address
Address relative to ImageBase (at ImageBase, RVA = 0)
Almost all addresses of the headers are RVAs
In code, addresses are *not* relative.

INT Import Name Table
Null-terminated list of pointers to Hint, Name structures

IAT Import Address Table
Null-terminated list of pointers
On file it is a copy of the INT
After loading it points to the imported APIs

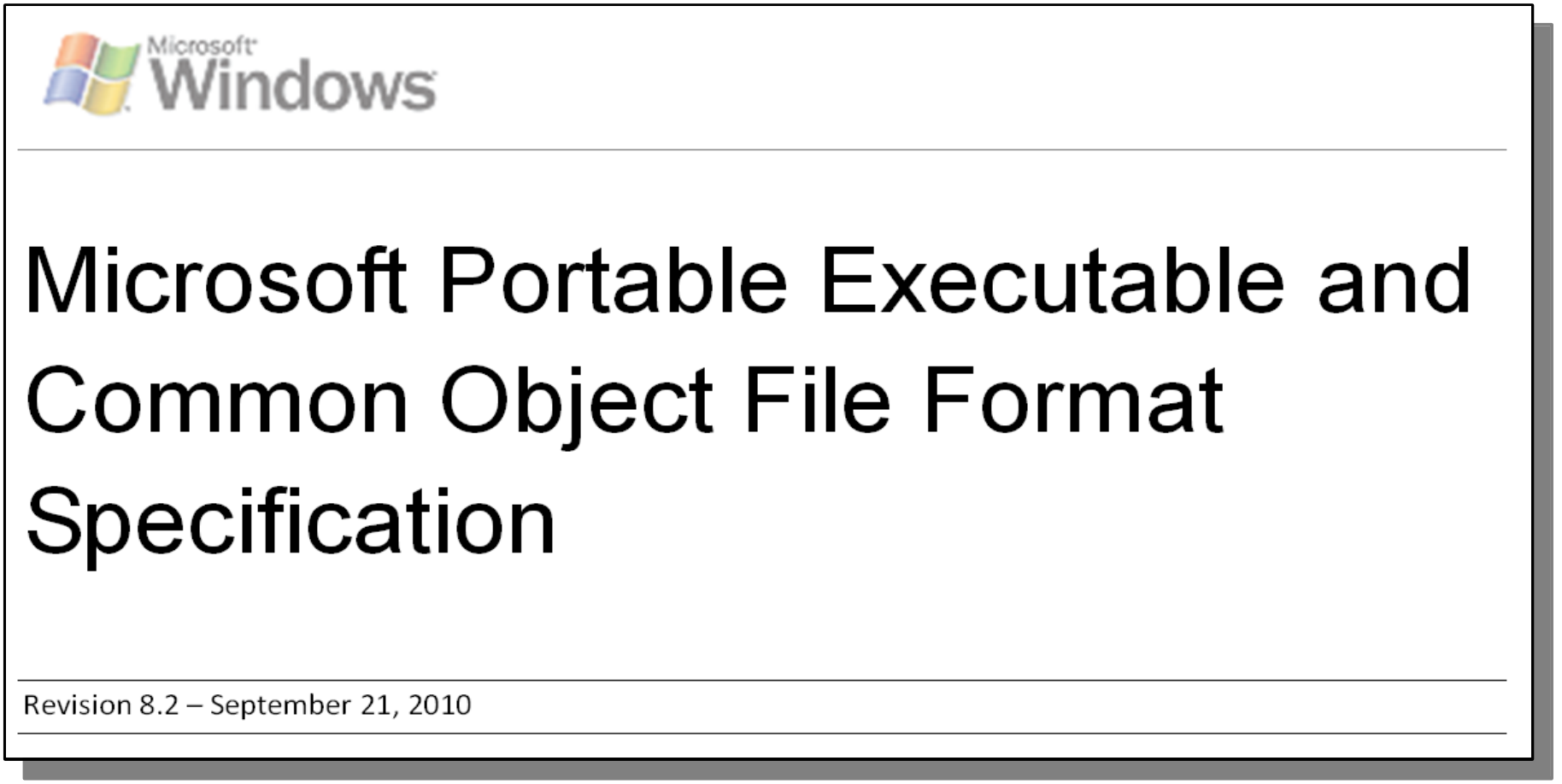
HINT

the problem...

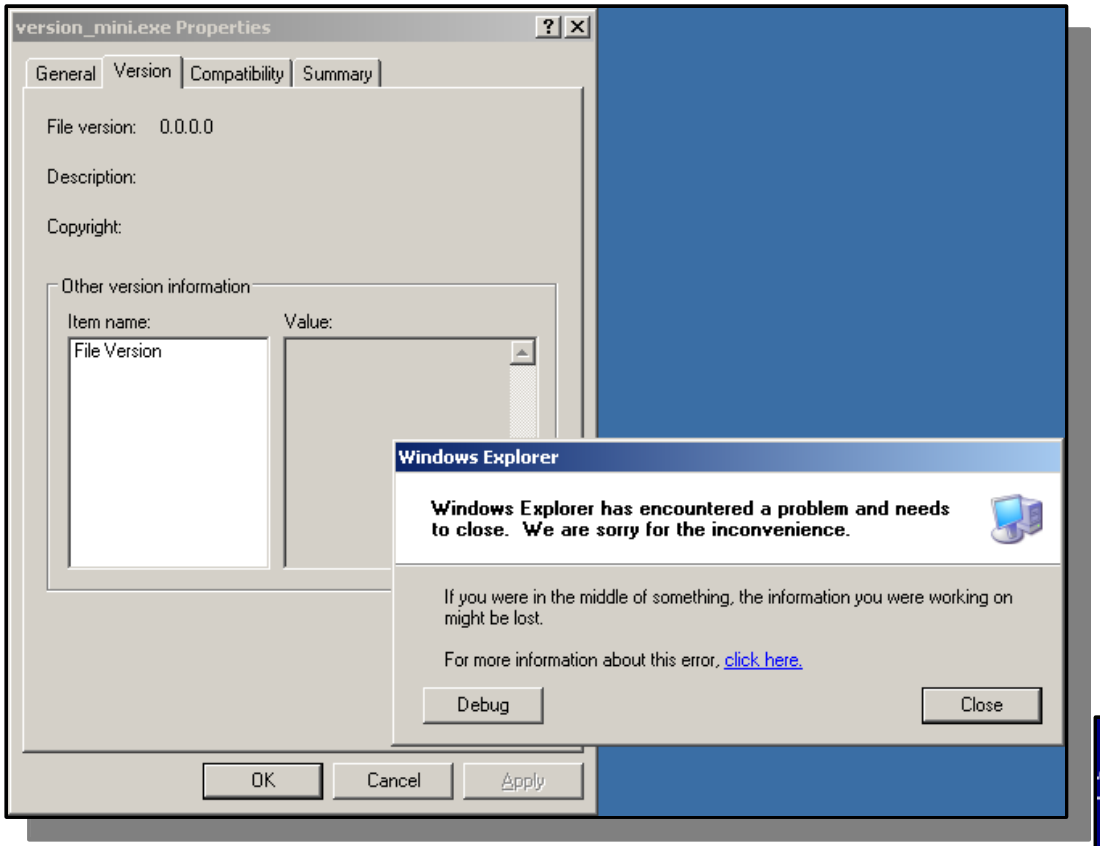
sins & punishments

- official documentation limited and unclear
  - just describes standard PEs
  - not good enough for security

- crashes (OS, security tools)
- obstacle for 3rd party developments
- hinders automation, classification
  - PE or not?
  - corrupted, or malware?
- fails best tools → prevents even manual analysis

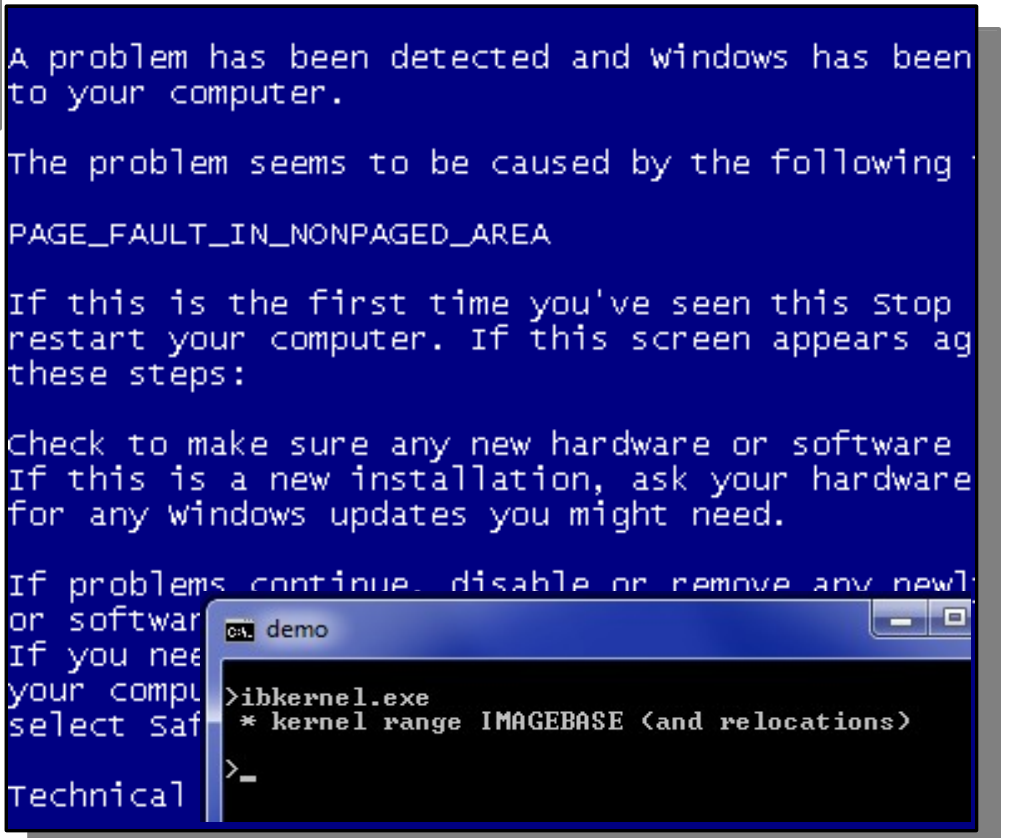


aka “the *gentle* guide to **standard** PEs”

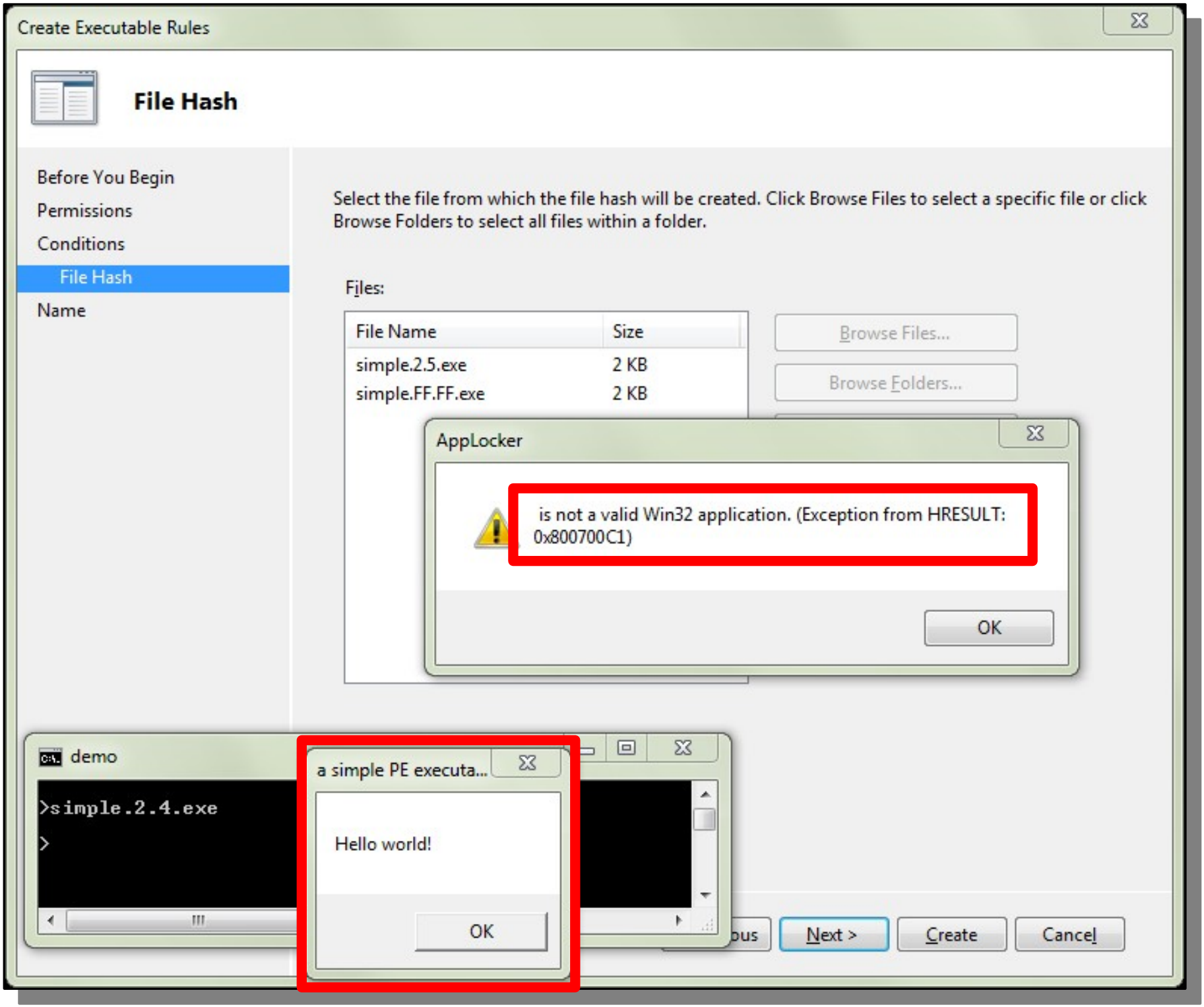


version_mini

CVE-2012-2273



ibkernel


Create Executable Rules
File Hash
Before You Begin
Permissions
Conditions
File Hash
Name
Select the file from which the file hash will be created. Click Browse Files to select a specific file or click Browse Folders to select all files within a folder.
Files:
File Name
Size
simple.2.5.exe
2 KB
simple.FF.FF.exe
2 KB
Browse Files...
Browse Folders...
AppLocker
is not a valid Win32 application. (Exception from HRESULT: 0x800700C1)
OK
demo
>simple.2.4.exe
>
a simple PE executa...
Hello world!
OK
Next >
Create
Cancel

normal

virustotal

SHA256: 8c9c4b0ff62552bb3ceca4016a5d729b7b42ca1897996b85a0c6362cfaa0e768

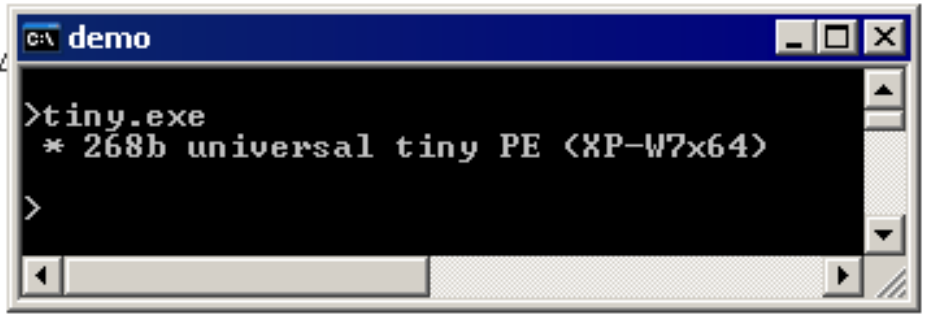
SHA1: cbe369c1881dca7e0848754b999c8840b5706260

MD5: 4fcef37d38a69c322994

File size: 268 bytes (268 bytes

File name: tiny.exe

File type: DOS EXE



SHA256: 2a9c7a16cdb3c3f2285afaf61072dd5e7cc022e97f351cad6234a13e5216f389

SHA1: e27faaa006229f8e4ab97fba7019dc9f2797f84d

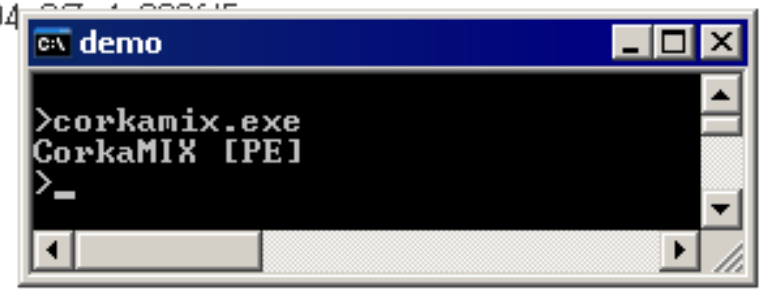
MD5: 88cad2b56ab67b43794

File size: 1.5 KB (1530 bytes)

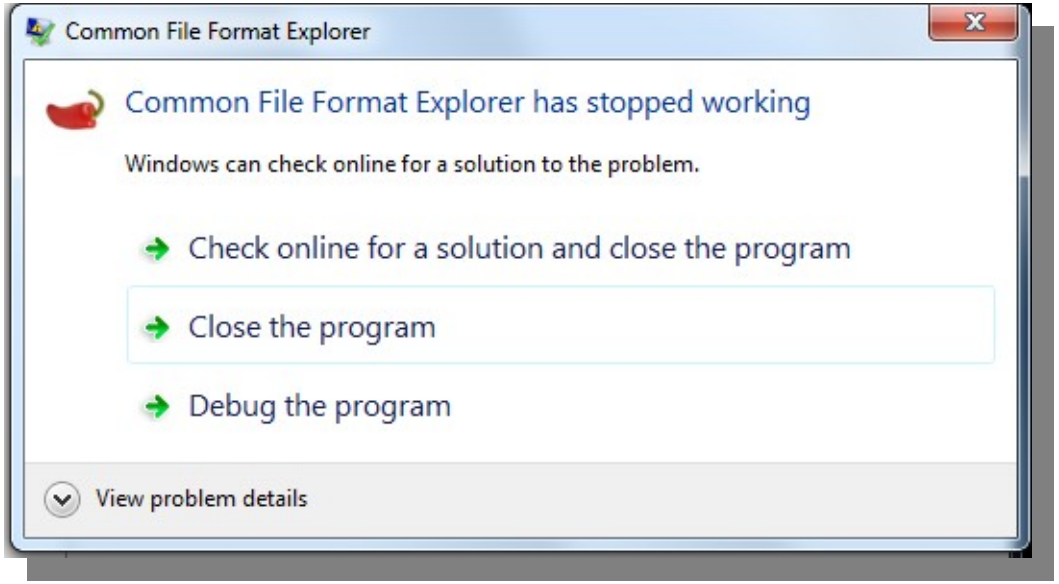
File name: corkamix.exe

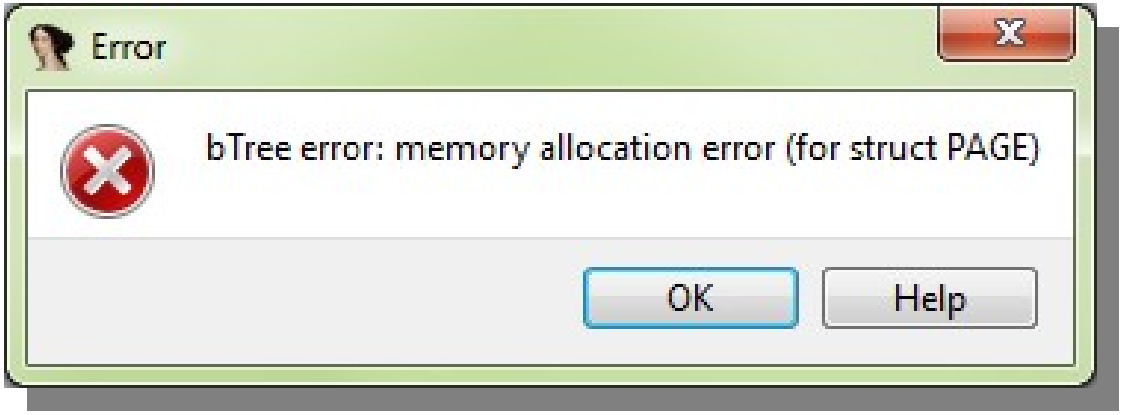
File type: PDF

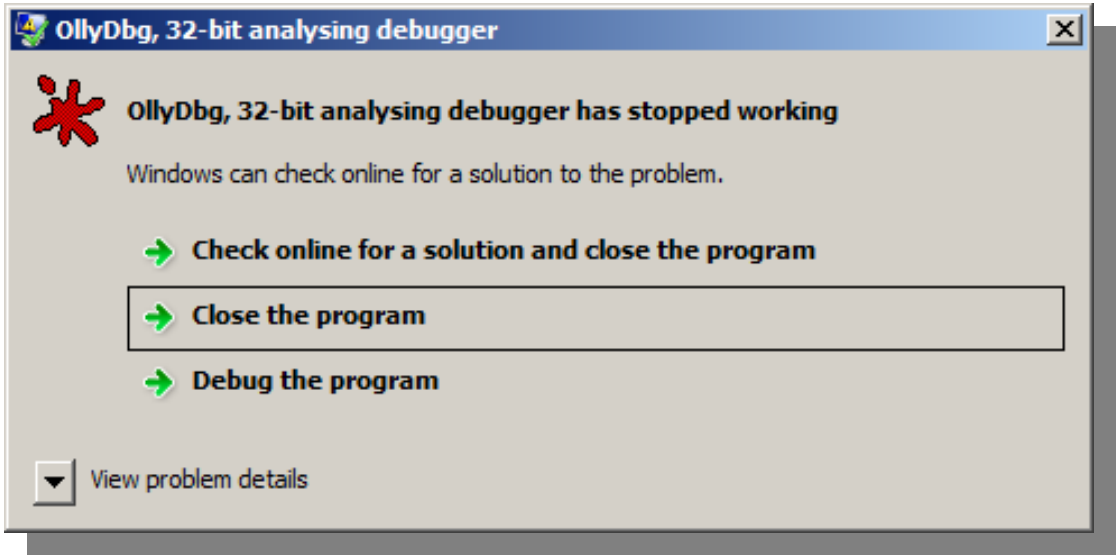
Tags: pdf



```
>cmp HelloWorld-upx.exe HelloWorld-upx-PATCHED.exe -b
HelloWorld-upx.exe HelloWorld-upx-PATCHED.exe differ: byte 481, line 4 is 125 U  40

>HelloWorld-upx-PATCHED.exe
Hello World!
>upx -d HelloWorld-upx.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2011
UPX 3.08w       Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     28160 <-      9728   34.55%    win32/pe     HelloWorld-upx.exe

Unpacked 1 file.

>upx -d HelloWorld-upx-PATCHED.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2011
UPX 3.08w       Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: HelloWorld-upx-PATCHED.exe: CantUnpackException: file is modified/hacked/protecte

Unpacked 0 files.
```


Common File Format Explorer
Common File Format Explorer has stopped working
Windows can check online for a solution to the problem.
Check online for a solution and close the program
Close the program
Debug the program
View problem details


Error
bTree error: memory allocation error (for struct PAGE)
OK
Help


OllyDbg, 32-bit analysing debugger
OllyDbg, 32-bit analysing debugger has stopped working
Windows can check online for a solution to the problem.
Check online for a solution and close the program
Close the program
Debug the program
View problem details

...and my approach

from bottom up

- ~~analyzing what's in the wild~~
  - waiting for malware/corruption to experiment?
- generate complete binaries from scratch
- manually
  - no framework/compiler limitation
  - concise PoCs

→ better coverage

I share knowledge and PoCs, with sources


RET
NOP
JMP
XOR
CALL
MOV
CMP
Int3
TEST
POP
Sub
ADD
PUSH
JZ
LEA


DOS_HEADER
MZ
NT_HEADERS
FILE HEADER
PE 1010
DATA DIRECTORY
102h
206h
106h

block by block

```
istruc IMAGE_DOS_HEADER
    at IMAGE_DOS_HEADER.e_magic,  db 'MZ'
    at IMAGE_DOS_HEADER.e_lfanew, dd NT_Signature - IMAGEBASE
iend
NT_Signature:
istruc IMAGE_NT_HEADERS
    at IMAGE_NT_HEADERS.Signature, db 'PE', 0, 0
iend
istruc IMAGE_FILE_HEADER
    at IMAGE_FILE_HEADER.Machine,         dw IMAGE_FILE_MACHINE_I386
    at IMAGE_FILE_HEADER.Characteristics, dw IMAGE_FILE_EXECUTABLE_IMAGE
iend
istruc IMAGE_OPTIONAL_HEADER32
```

a *complete* executable


```
IMAGEBASE equ 400000h
org IMAGEBASE

istruc IMAGE_DOS_HEADER
    at IMAGE_DOS_HEADER.e_magic,  db 'MZ'
    at IMAGE_DOS_HEADER.e_lfanew, dd NT_Signature - IMAGEBASE
iend
NT_Signature:
istruc IMAGE_NT_HEADERS
    at IMAGE_NT_HEADERS.Signature, db 'PE', 0, 0
iend
istruc IMAGE_FILE_HEADER
    at IMAGE_FILE_HEADER.Machine,         dw IMAGE_FILE_MACHINE_I386
    at IMAGE_FILE_HEADER.Characteristics, dw IMAGE_FILE_EXECUTABLE_IMAGE
iend
istruc IMAGE_OPTIONAL_HEADER32
    at IMAGE_OPTIONAL_HEADER32.Magic,                  dw IMAGE_NT_OPTIONAL_HDR32_MAGIC
    at IMAGE_OPTIONAL_HEADER32.AddressOfEntryPoint,    dd EntryPoint - IMAGEBASE ; not strictly required
    at IMAGE_OPTIONAL_HEADER32.ImageBase,              dd IMAGEBASE ; not required under XP
    at IMAGE_OPTIONAL_HEADER32.SectionAlignment,       dd 1
    at IMAGE_OPTIONAL_HEADER32.FileAlignment,          dd 1
    at IMAGE_OPTIONAL_HEADER32.MajorSubsystemVersion, dw 4
    at IMAGE_OPTIONAL_HEADER32.SizeOfImage,            dd SIZEOFIMAGE
    at IMAGE_OPTIONAL_HEADER32.SizeOfHeaders,          dd SIZEOFIMAGE - 1 ; required for XP
    at IMAGE_OPTIONAL_HEADER32.Subsystem,              dw IMAGE_SUBSYSTEM_WINDOWS_CUI
iend
istruc IMAGE_DATA_DIRECTORY_16
iend

EntryPoint:
    push 42
    pop eax
    retn
```

PE

pe.corkami.com

PointerToSymbolTable/NumberOfSymbols

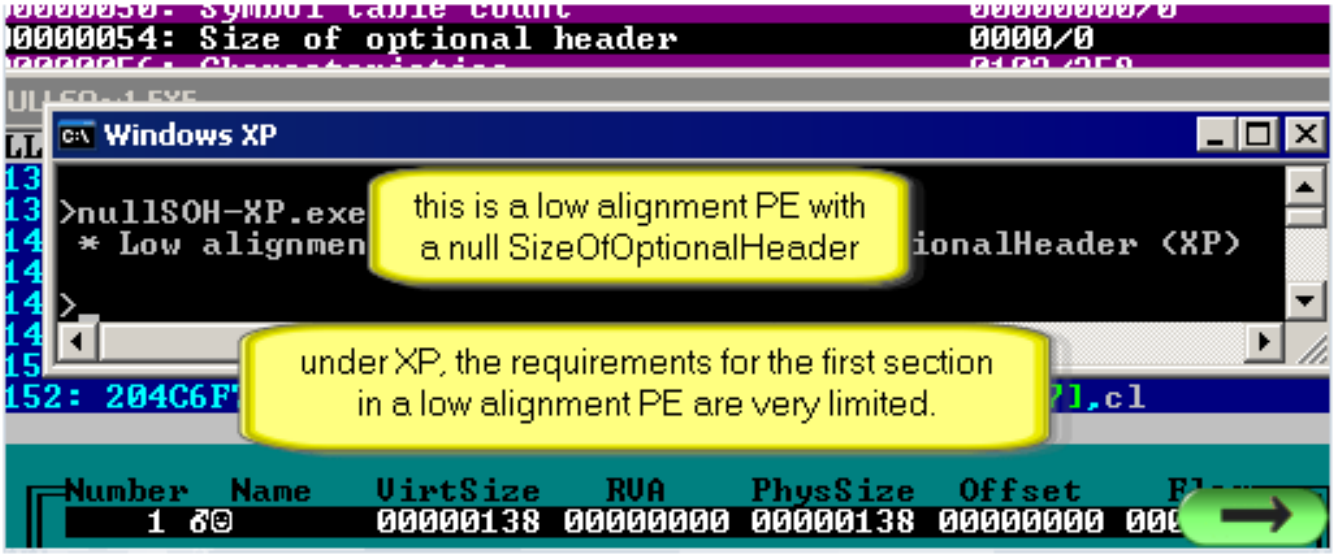
no importance whatsoever for the loader

SizeOfOptionalHeader

- is not the size of the optional header, but the delta betwe

  Thus, it can be null (the section table will overlap the Opt
  negative.

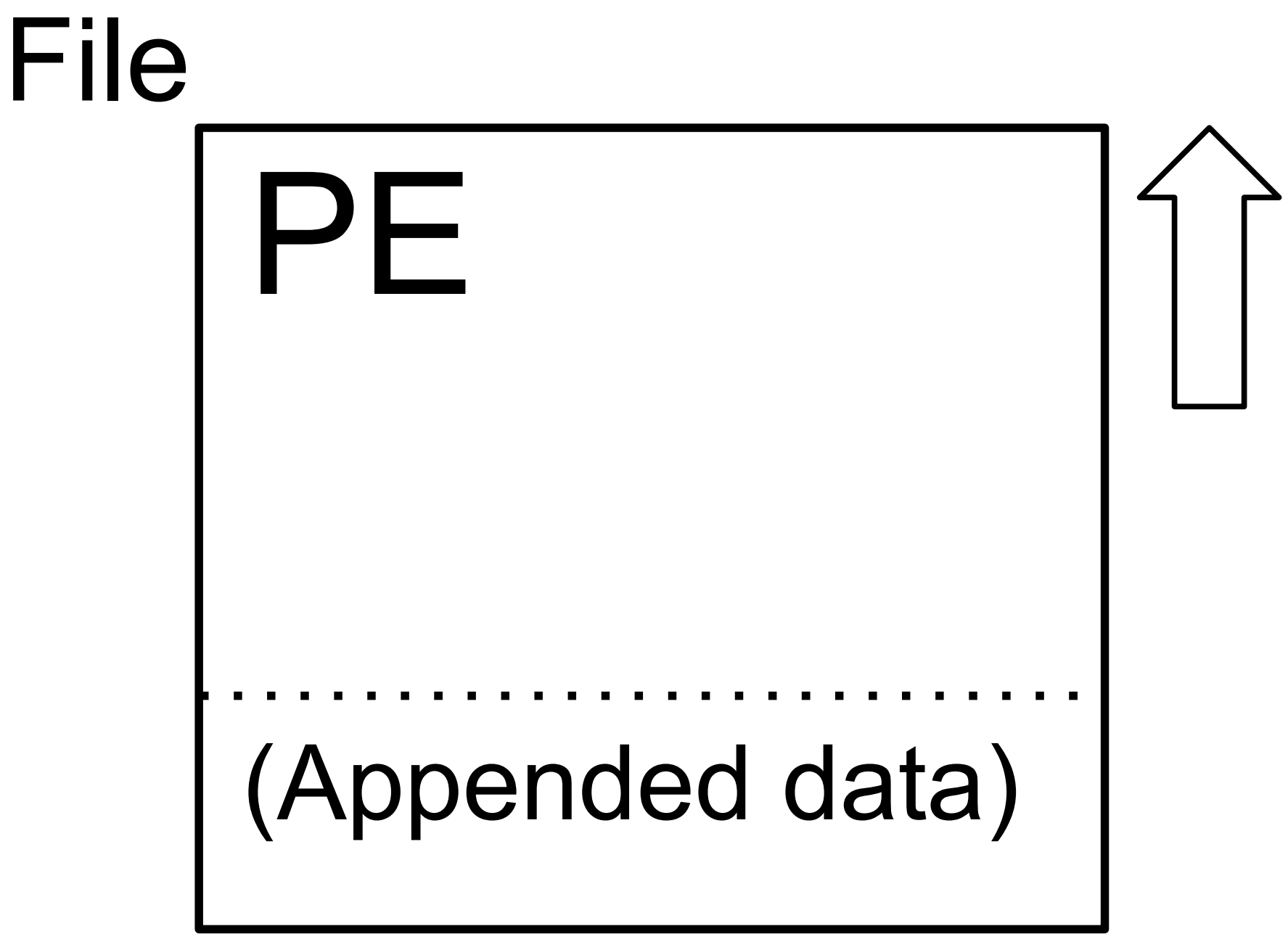
  - `nullSOH-XP` is a PE with a null SizeOfHeaders. the section table is thus overlapping the optio

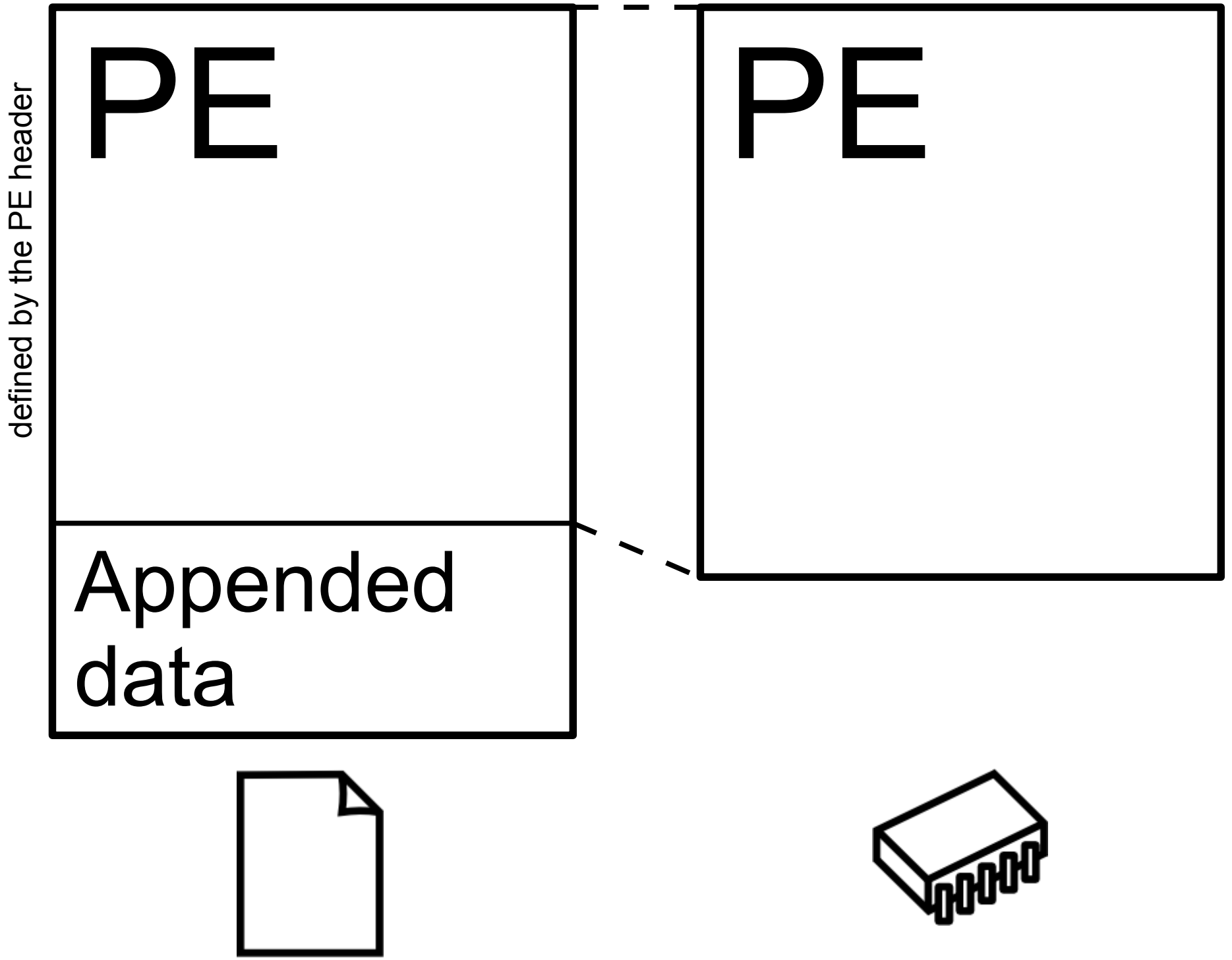



```
...
SECTIONALIGN equ 4
FILEALIGN equ 4
...
OptionalHeader:
```

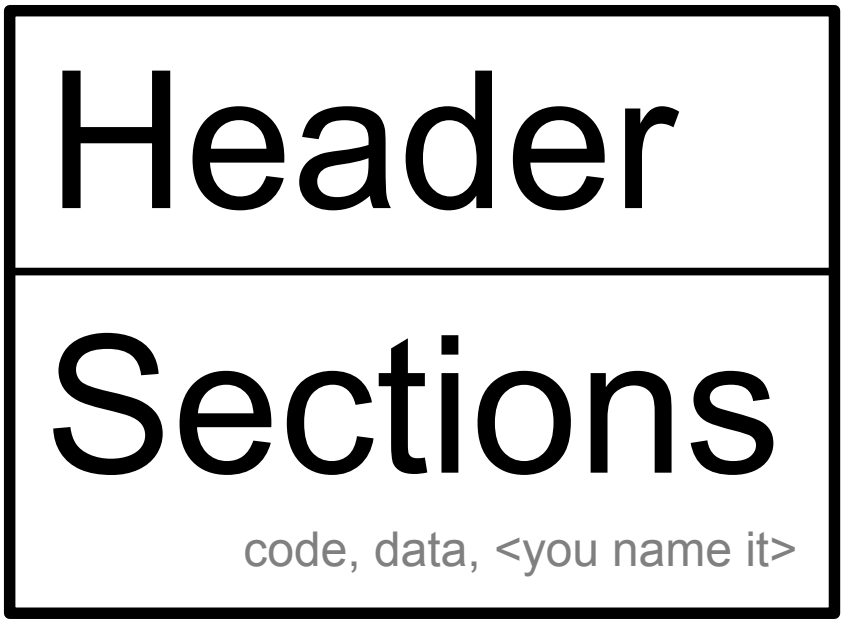
pe.corkami.com


le
PE

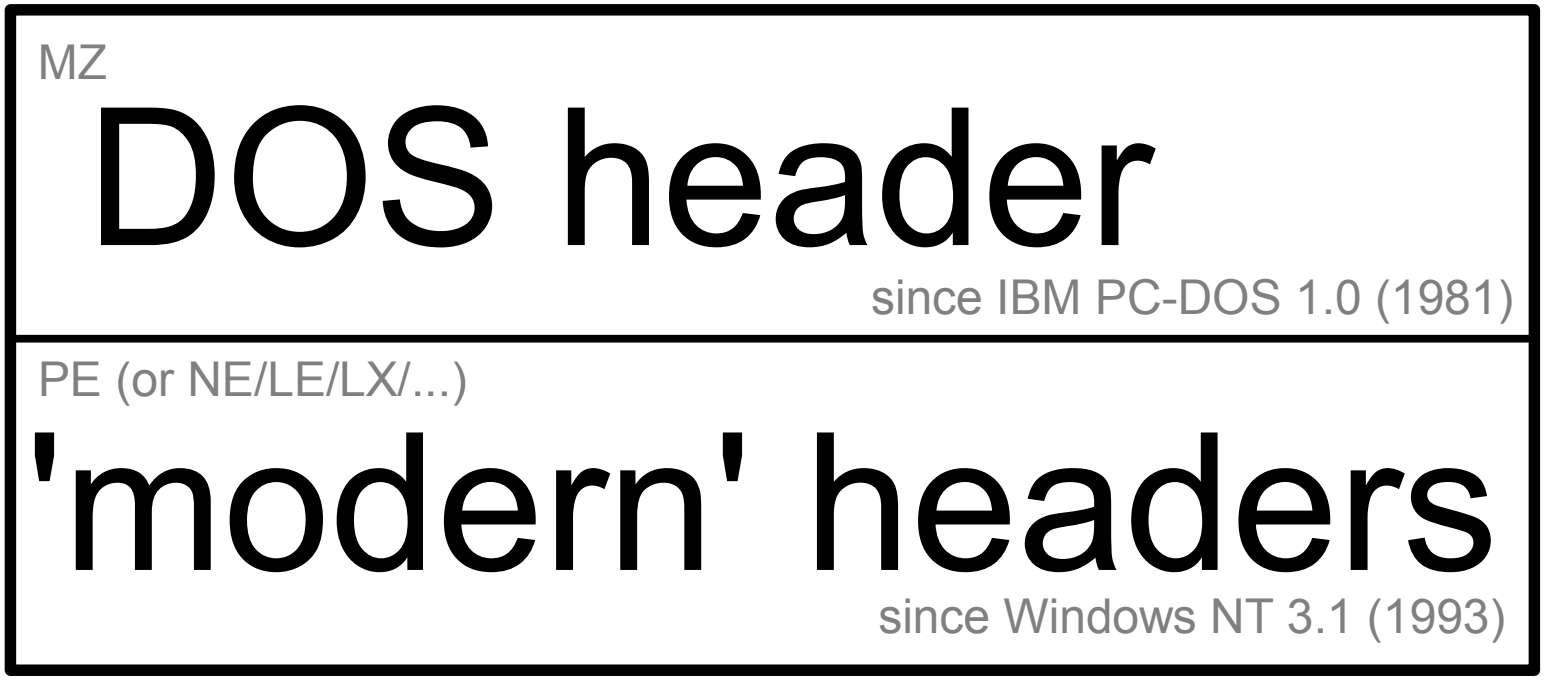

File
PE
(Appended data)


PE
PE
defined by the PE header
Appended
data

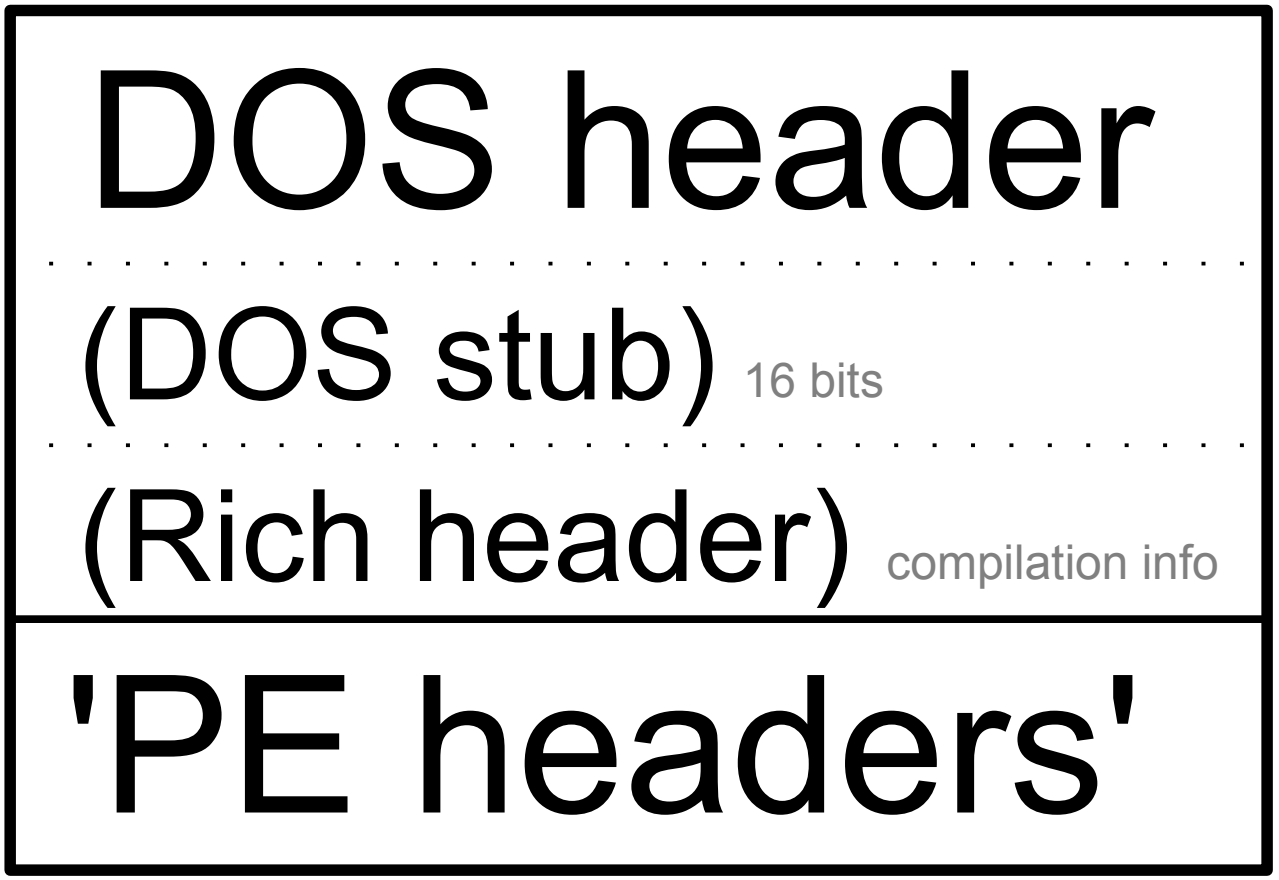
PE



Header


MZ
DOS header
since IBM PC-DOS 1.0 (1981)
PE (or NE/LE/LX/...)
'modern' headers
since Windows NT 3.1 (1993)

Header


DOS header
(DOS stub) 16 bits
(Rich header) compilation info
'PE headers'

DOS Stub

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	♫ º♫.´.Í!¸☺LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is.program.canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t.be.run.in.DOS.
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.......

```
start:
    push    cs
    pop     ds
    mov     dx, msg - start
    mov     ah, 9
    int     21h     ; print string

    mov     ax, 4c01h
    int     21h     ; exit
msg:
    db 'This program cannot be run in DOS mode.', 0dh, 0dh, 0ah, '$'
```

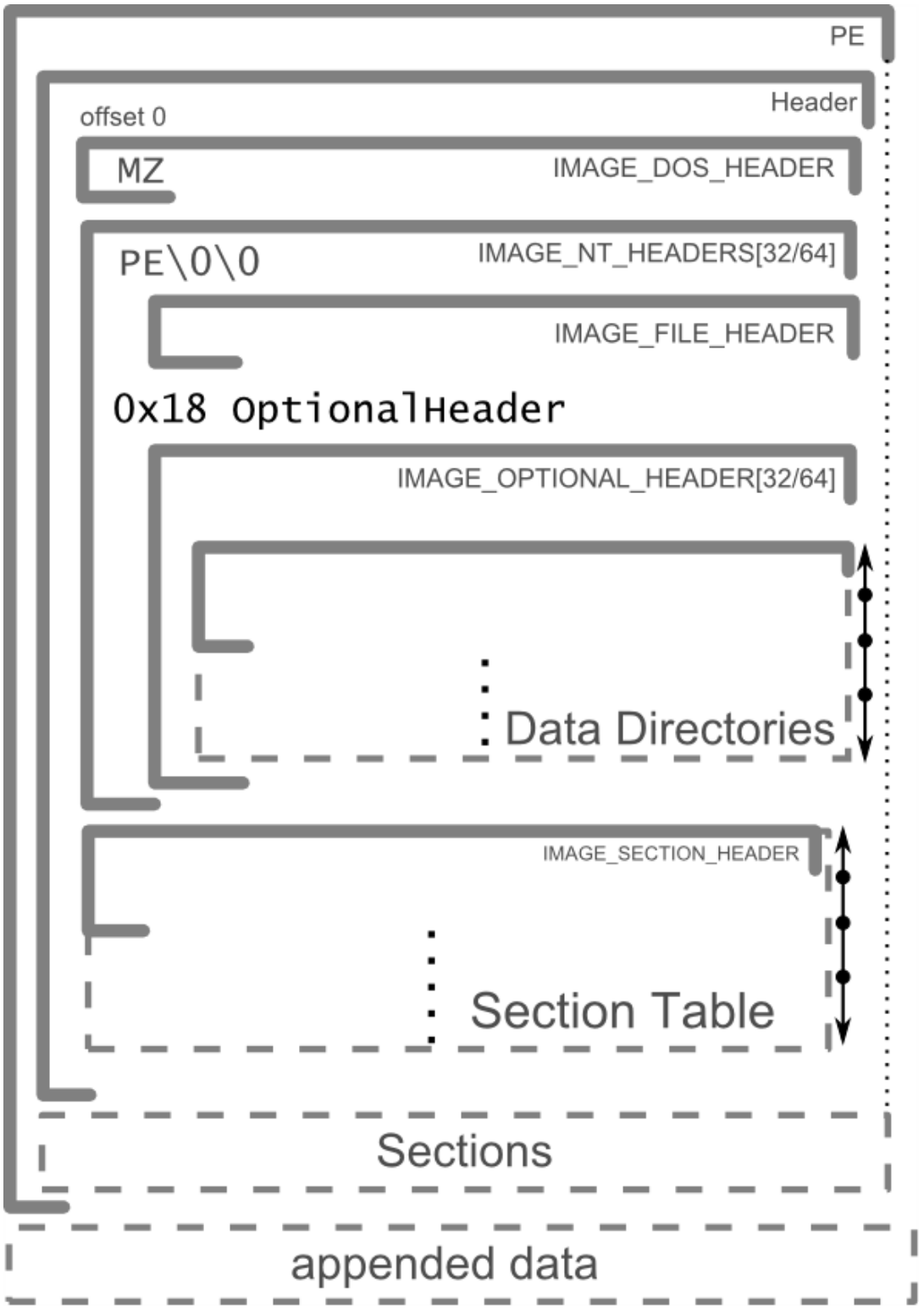
- obsolete 16b code
  - prints msg & exits
- still present on all standard PEs
  - even 64b binaries

PoC: compiled

'Rich' header

- compiler information
- officially undocumented
  - pitiful xor32 encryption
- completely documented by Daniel Pistelli
  http://ntcore.com/files/richsign.htm

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000080	EC	85	5B	A1	A8	E4	35	F2	A8	E4	35	F2	A8	E4	35	F2	ì▌[¡¨ä5ò¨ä5ò¨ä5ò
00000090	6B	EB	3A	F2	A9	E4	35	F2	6B	EB	55	F2	A9	E4	35	F2	kë:ò©ä5òkëUò©ä5ò
000000A0	6B	EB	68	F2	BB	E4	35	F2	A8	E4	34	F2	63	E4	35	F2	këhò»ä5ò¨ä4òcä5ò
000000B0	6B	EB	6B	F2	A9	E4	35	F2	6B	EB	6A	F2	BF	E4	35	F2	këkò©ä5òkëjò¿ä5ò
000000C0	6B	EB	6F	F2	A9	E4	35	F2	52	69	63	68	A8	E4	35	F2	këoò©ä5òRich¨ä5ò
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....


PE
Header
offset 0
MZ
IMAGE_DOS_HEADER
PE\0\0
IMAGE_NT_HEADERS[32/64]
IMAGE_FILE_HEADER
0x18 OptionalHeader
IMAGE_OPTIONAL_HEADER[32/64]
Data Directories
IMAGE_SECTION_HEADER
Section Table
Sections
appended data

Dos header

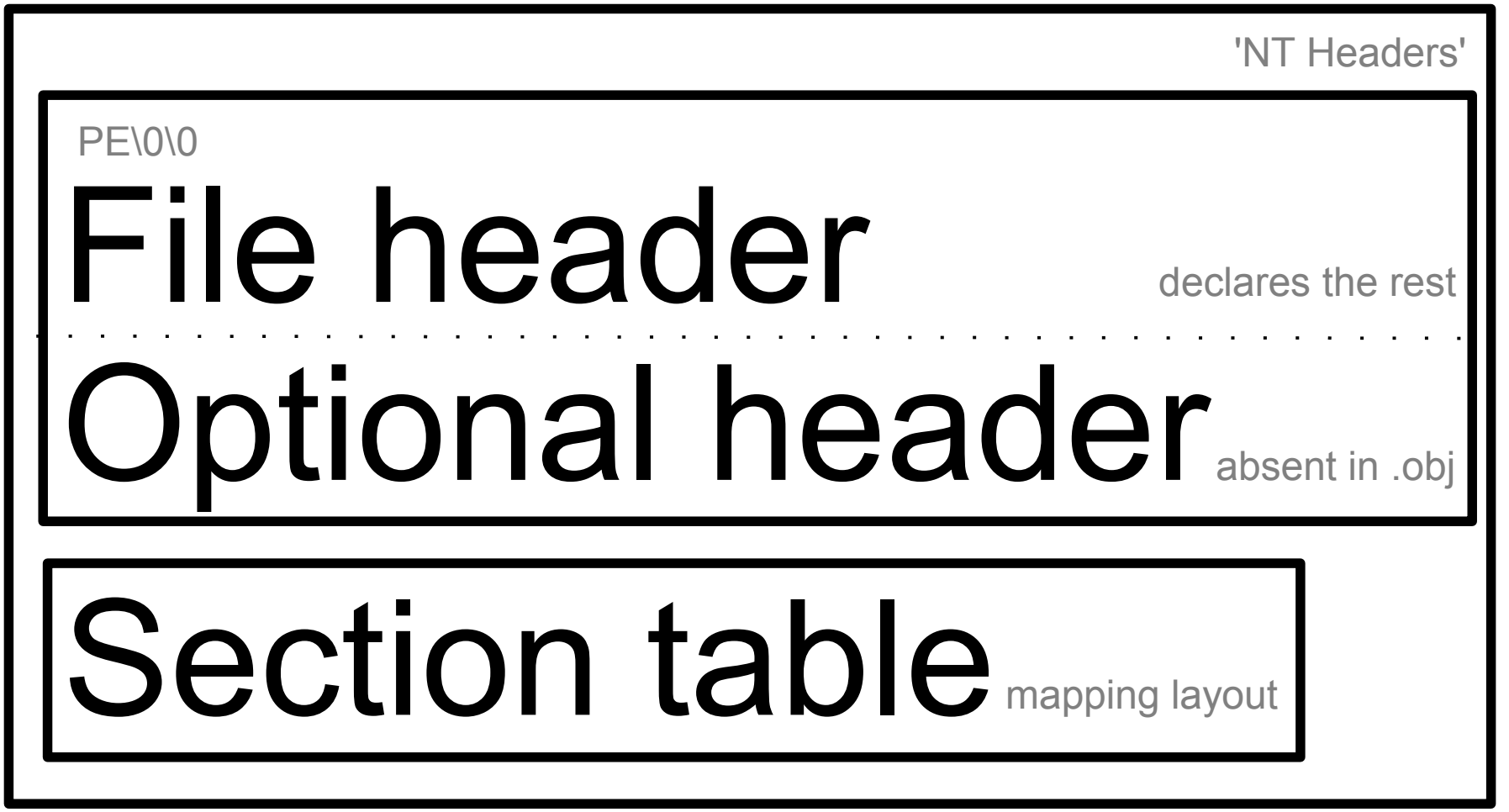
- obsolete stuff
  - only used if started in DOS mode
  - ignored otherwise
- tells where the PE header is

offset 0

IMAGE_DOS_HEADER

```
0x00 dw e_magic MZ
0x02 dw e_cblp
0x04 dw e_cp       exe size
0x06 dw e_crlc
0x08 dw e_cparhdr exe start
0x0a dw e_minalloc
0x0c dw e_maxalloc
0x0e dw e_ss
0x10 dw e_sp
0x12 dw e_csum
0x14 dw e_ip
0x16 dw e_cs
0x18 dw e_lfarlc
0x1a dw e_ovno
0x1c dw e_res[4]
0x24 dw e_oemid
0x26 dw e_oeminfo
0x28 dw e_res2[10]
0x3c dd e_lfanew ——————> offset to PE Header
```

'PE Headers'


'NT Headers'
PE\0\0
File header
declares the rest
Optional header
absent in .obj
Section table
mapping layout

File header

- how many sections?
- is there an Optional Header?
- 32b or 64b, DLL or EXE...

IMAGE_FILE_HEADER

0x00 dw Machine 0x014c [32b]/0x8664 [64b]

0x02 dw NumberOfSections

0x04 dd TimeDateStamp

0x08 dd PointerToSymbolTable

0x0c dd NumberOfSymbols

0x10 dw SizeOfOptionalHeader

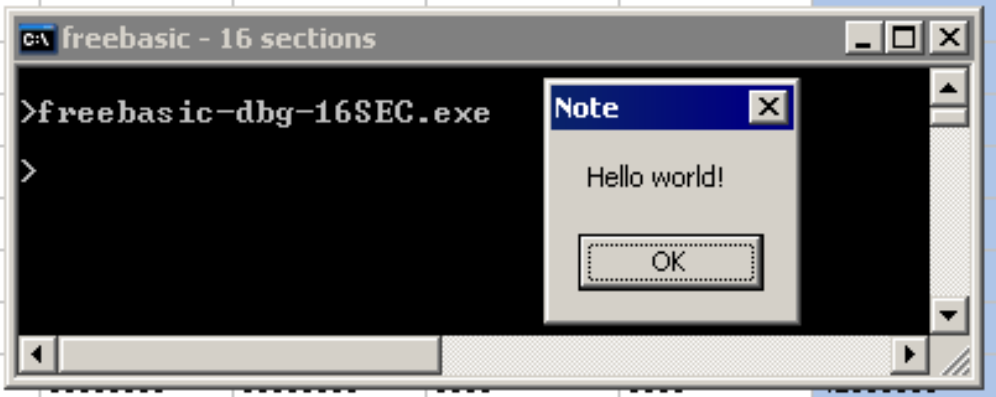
0x12 dw Characteristics exe/dll,relocs

NumberOfSections values

- 0: Corkami :p
- 1: packer
- 3-6: standard
  - code, data, (un)initialized data, imports, resources...

- 16: free basic FTW :D
  - what for ?

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00000C30	00001000	00000E00	00000600	00000000	00000000	0000	0000	60000060
.data	0000002C	00002000	00000200	00001400	00000000	00000000	0000	0000	C0000040
.jcr	00000004	00003000	00000200	00001600					
.rdata	0000022C	00004000	00000400	00001800					
.bss	00004194	00005000	00000000	00000000					
.idata	000003B0	0000A000	00000400	00001C00					
.stab	00000174	0000B000	00000200	00002000					
.stabstr	00000160	0000C000	00000200	00002200					
/4	00000040	0000D000	00000200	00002400					
/19	000000F4	0000E000	00000200	00002600					
/35	00000DB9	0000F000	00000E00	00002800	00000000	00000000	0000	0000	42000000
/47	00000340	00010000	00000400	00003600	00000000	00000000	0000	0000	42000000
/61	0000025F	00011000	00000400	00003A00	00000000	00000000	0000	0000	42000000
/73	000000E0	00012000	00000200	00003E00	00000000	00000000	0000	0000	42000000
/86	0000026E	00013000	00000400	00004000	00000000	00000000	0000	0000	42000000
/97	00000030	00014000	00000200	00004400	00000000	00000000	0000	0000	42000000



Optional header

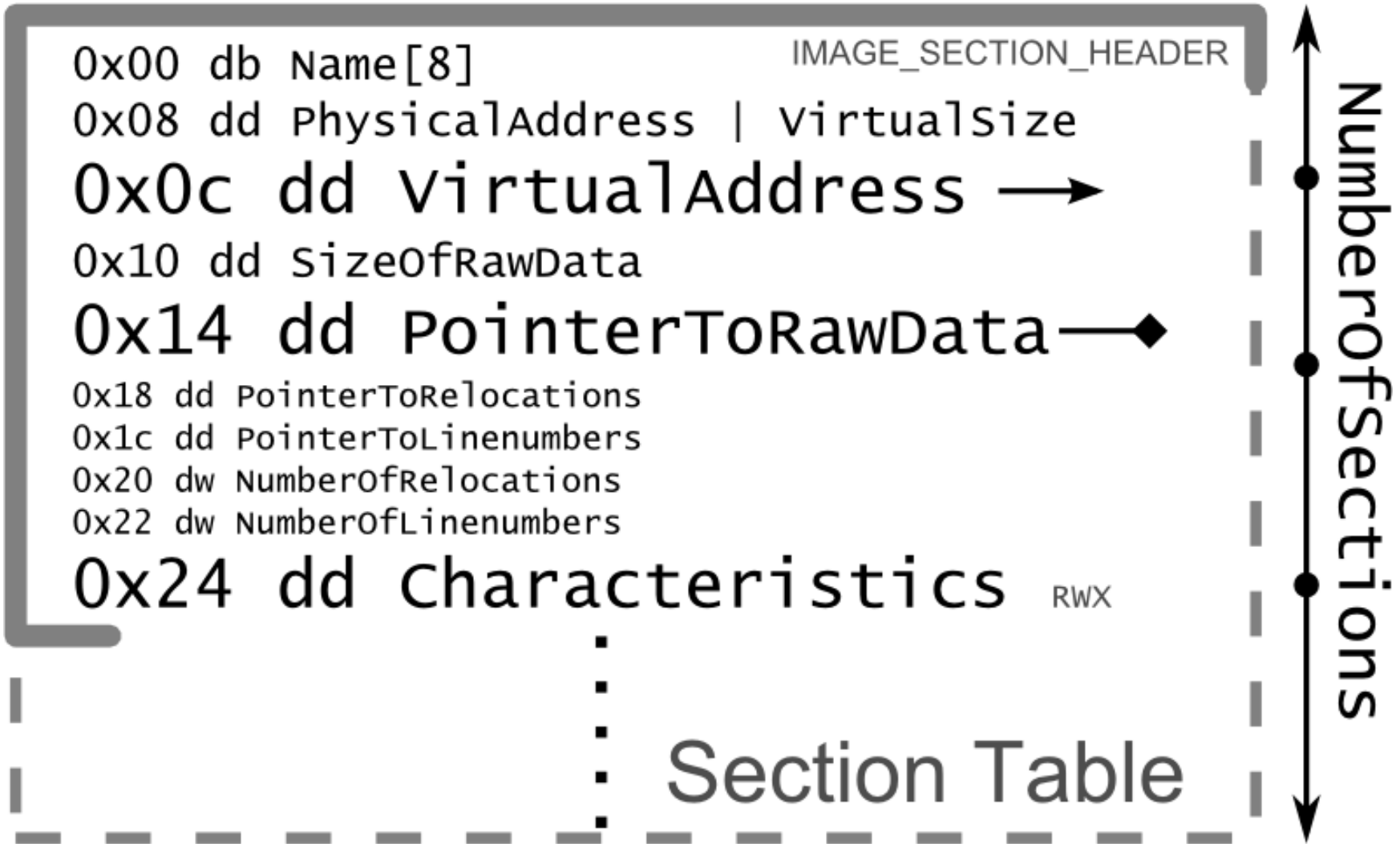
- geometry properties
  - alignments, base, size
- tells where code starts
- 32/64b, driver/standard/console
- many non critical information
- data directory

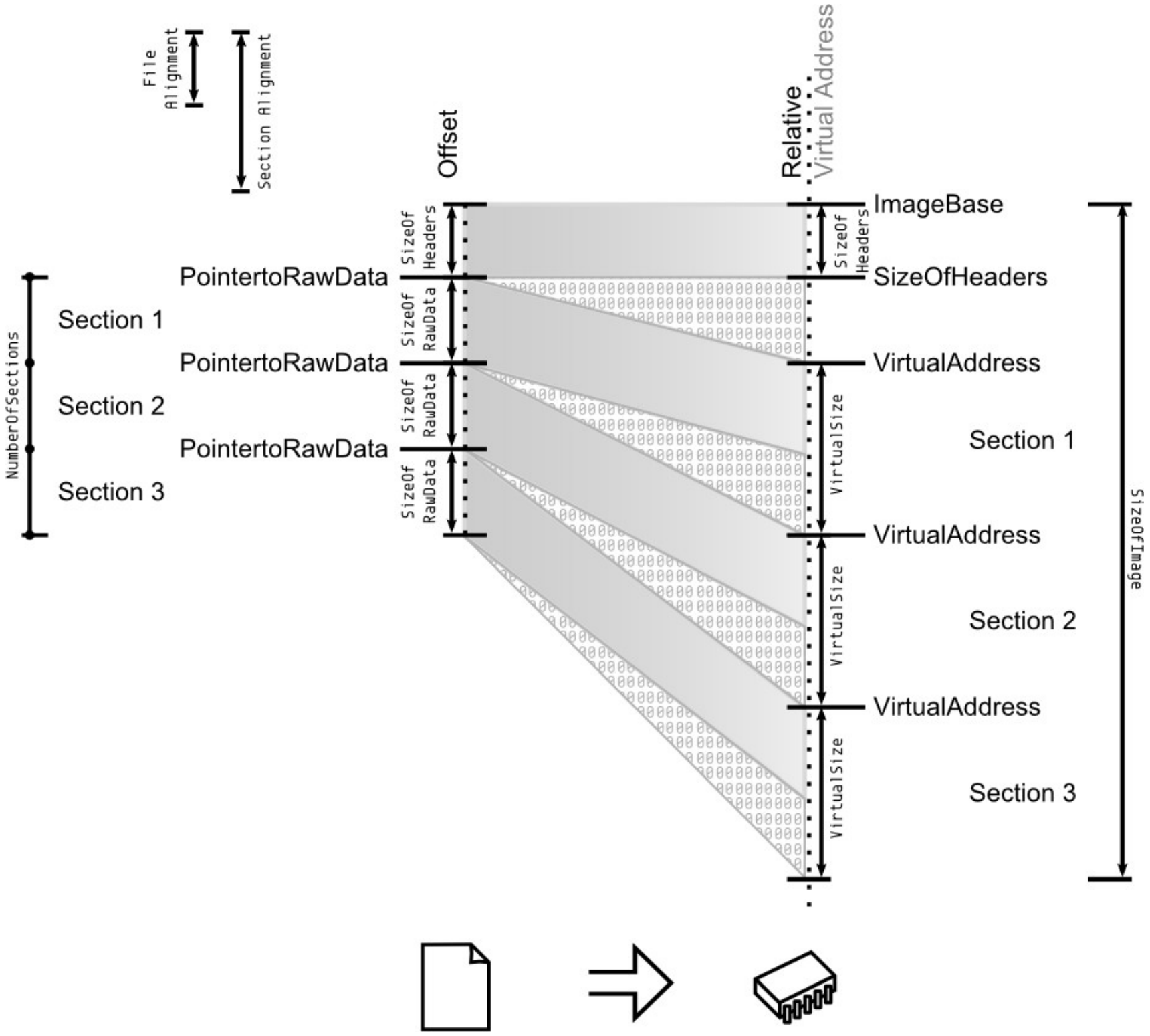
IMAGE_OPTIONAL_HEADER[32/64]

0x00 dw Magic 0x10b [32b]/0x20b [64b]
0x02 db MajorLinkerVersion
0x03 db MinorLinkerVersion
0x04 dd SizeOfCode
0x08 dd SizeOfInitializedData
0x0c dd SizeOfUninitializedData
0x10 dd AddressOfEntryPoint →
0x14 dd BaseOfCode dq in 64b →
0x18 dd BaseOfData only in 32b →
0x1c dd ImageBase dq in 64b
0x20 dd SectionAlignment =2^y, with y≥x
0x24 dd FileAlignment =2^x
0x28 dw MajorOperatingSystemVersion 4/5
0x2a dw MinorOperatingSystemVersion
0x2c dw MajorImageVersion
0x2e dw MinorImageVersion
0x30 dw MajorSubsystemVersion
0x32 dw MinorSubsystemVersion
0x34 dd Win32VersionValue
0x38 dd SizeOfImage
0x3c dd SizeOfHeaders
0x40 dd CheckSum [drivers]
0x44 dw Subsystem 1 driver/2 gui/3 cli
0x46 dw DllCharacteristics
0x48 dd SizeOfStackReserve dq in 64b
0x4c dd SizeOfStackCommit dq in 64b
0x50 dd SizeOfHeapReserve dq in 64b
0x54 dd SizeOfHeapCommit dq in 64b
0x58 dd LoaderFlags
0x5c dd NumberOfRvaAndSizes ≤16
0x60 DataDirectory[]

Sections

- defines the mapping:
  - which part of the file goes where
  - what for? (writeable, executable...)


IMAGE_SECTION_HEADER
0x00 db Name[8]
0x08 dd PhysicalAddress | VirtualSize
0x0c dd VirtualAddress
0x10 dd SizeOfRawData
0x14 dd PointerToRawData
0x18 dd PointerToRelocations
0x1c dd PointerToLinenumbers
0x20 dw NumberOfRelocations
0x22 dw NumberOfLinenumbers
0x24 dd Characteristics
RWX
Section Table
NumberOfSections


File Alignment
Section Alignment
Offset
Relative Virtual Address
NumberOfSections
Section 1
Section 2
Section 3
PointertoRawData
PointertoRawData
PointertoRawData
SizeOf Headers
SizeOf RawData
SizeOf RawData
SizeOf RawData
ImageBase
SizeOfHeaders
VirtualAddress
VirtualAddress
VirtualAddress
VirtualSize
VirtualSize
VirtualSize
Section 1
Section 2
Section 3
SizeOfImage

Data Directory

- (RVA, Size) DataDirectory[NumbersOfRvaAndSizes]
- each of the standard 16 firsts has a specific use

  → often called 'Data Directories'

IMAGE_DATA_DIRECTORY[]

0 IMAGE_DIRECTORY_ENTRY_EXPORT provide external APIs

1 IMAGE_DIRECTORY_ENTRY_IMPORT request external APIs

2 IMAGE_DIRECTORY_ENTRY_RESOURCE icons/manifest/version/...

3 IMAGE_DIRECTORY_ENTRY_SECURITY signing certificate

4 IMAGE_DIRECTORY_ENTRY_EXCEPTION 64b SEH

5 IMAGE_DIRECTORY_ENTRY_BASERELOC relocations

6 IMAGE_DIRECTORY_ENTRY_DEBUG debug symbols

7 IMAGE_DIRECTORY_ENTRY_COPYRIGHT useless

8 IMAGE_DIRECTORY_ENTRY_GLOBALPTR obsolete

9 IMAGE_DIRECTORY_ENTRY_TLS called before Entrypoint

A IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG SafeSEH

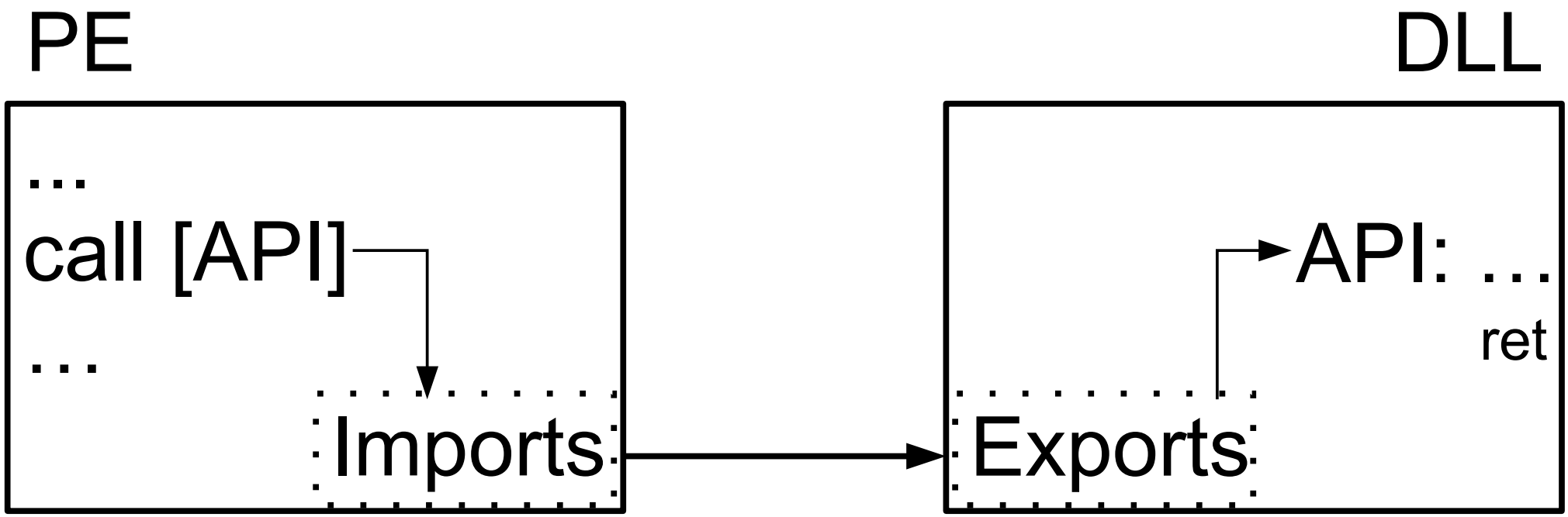
B IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT speed up imports

C IMAGE_DIRECTORY_ENTRY_IAT where imports are writeable

D IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT

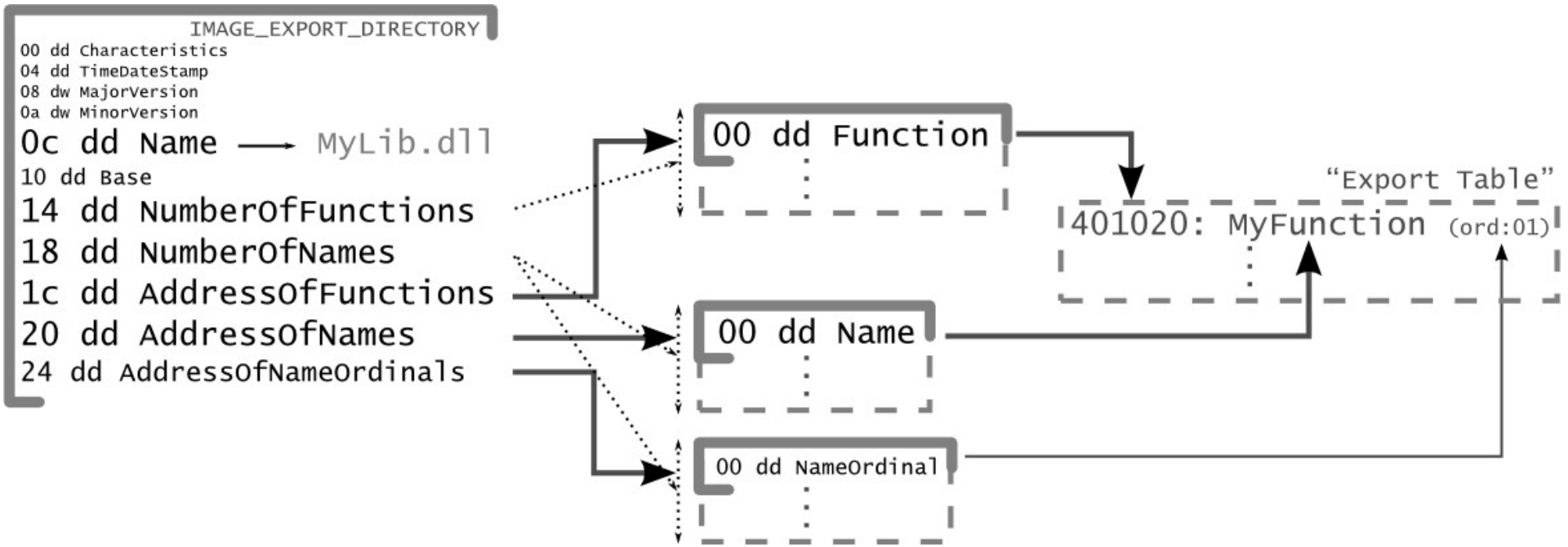
E IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR .Net header

F reserved unused


PE
DLL
...
call [API]
...
Imports
Exports
API: ...
ret

Exports

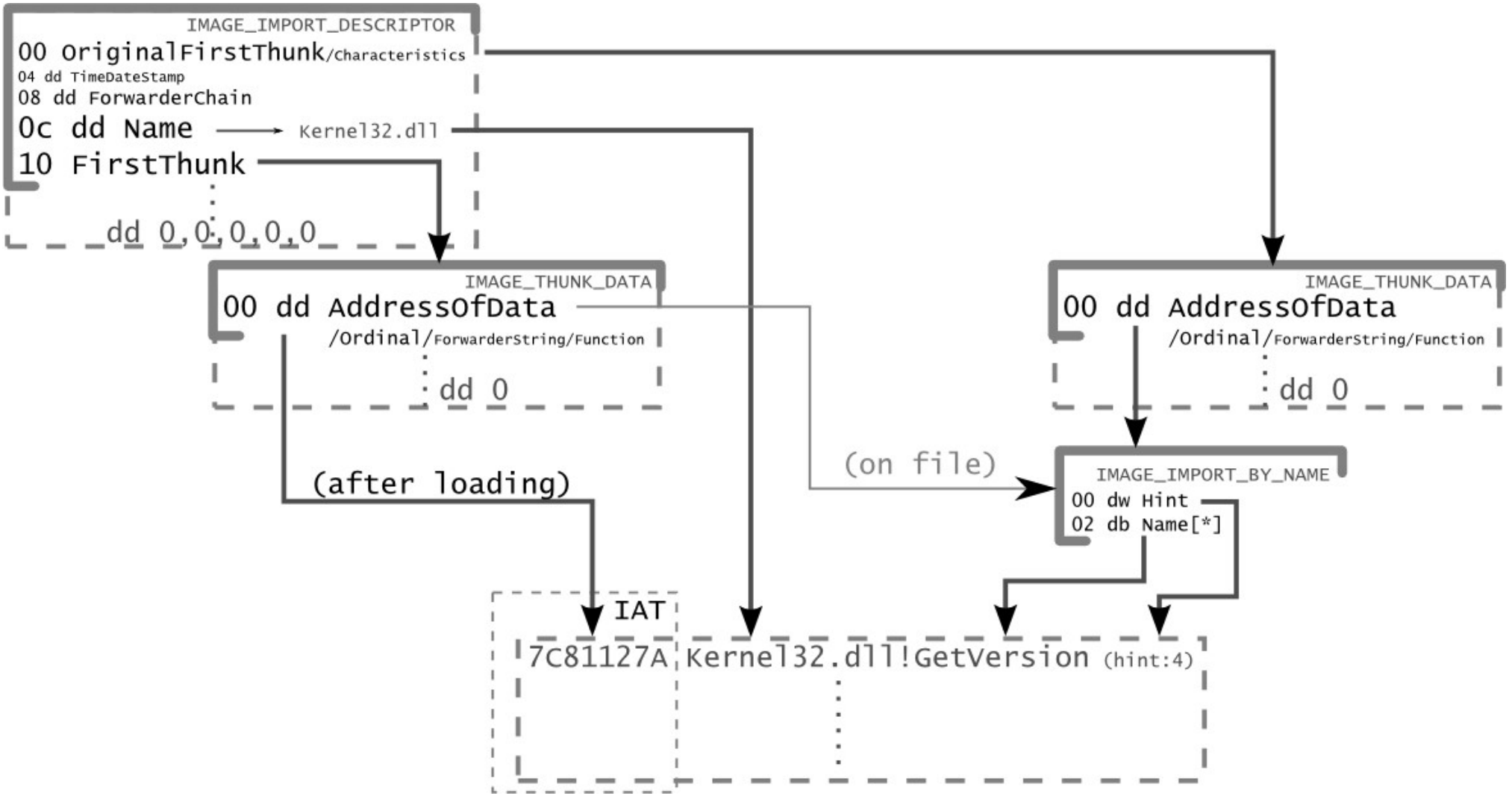
- 3 pointers to 3 lists
- defining in parallel (name, address, ordinal)
  - a function can have several names


IMAGE_EXPORT_DIRECTORY
00 dd Characteristics
04 dd TimeDateStamp
08 dw MajorVersion
0a dw MinorVersion
0c dd Name
MyLib.dll
10 dd Base
14 dd NumberOfFunctions
18 dd NumberOfNames
1c dd AddressOfFunctions
20 dd AddressOfNames
24 dd AddressOfNameOrdinals
00 dd Function
00 dd Name
00 dd NameOrdinal
"Export Table"
401020: MyFunction (ord:01)

Imports

- a null-terminated list of descriptors
  - typically one per imported DLL
- each descriptor specifies
  - DLL's name
  - 2 null-terminated lists of pointers
    - API names and future API addresses

- ImportsAddressTable highlights the address table
  - for write access

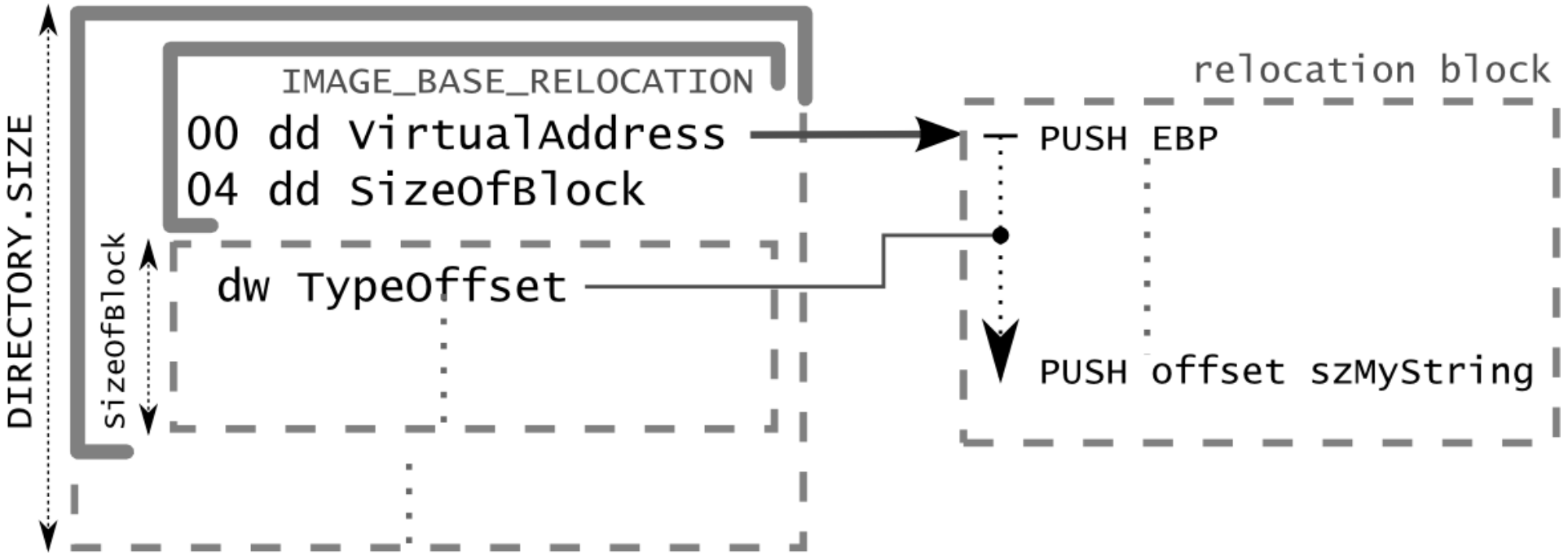

IMAGE_IMPORT_DESCRIPTOR
00 OriginalFirstThunk/Characteristics
04 dd TimeDateStamp
08 dd ForwarderChain
0c dd Name
Kernel32.dll
10 FirstThunk
dd 0,0,0,0,0
IMAGE_THUNK_DATA
00 dd AddressOfData
/Ordinal/ForwarderString/Function
dd 0
IMAGE_THUNK_DATA
00 dd AddressOfData
/Ordinal/ForwarderString/Function
dd 0
(on file)
IMAGE_IMPORT_BY_NAME
00 dw Hint
02 db Name[*]
(after loading)
IAT
7C81127A
Kernel32.dll!GetVersion (hint:4)

Relocations

- PE have standard ImageBases
  - EXE: 0x400000, DLL 0x1000000

    → conflicts between DLLs

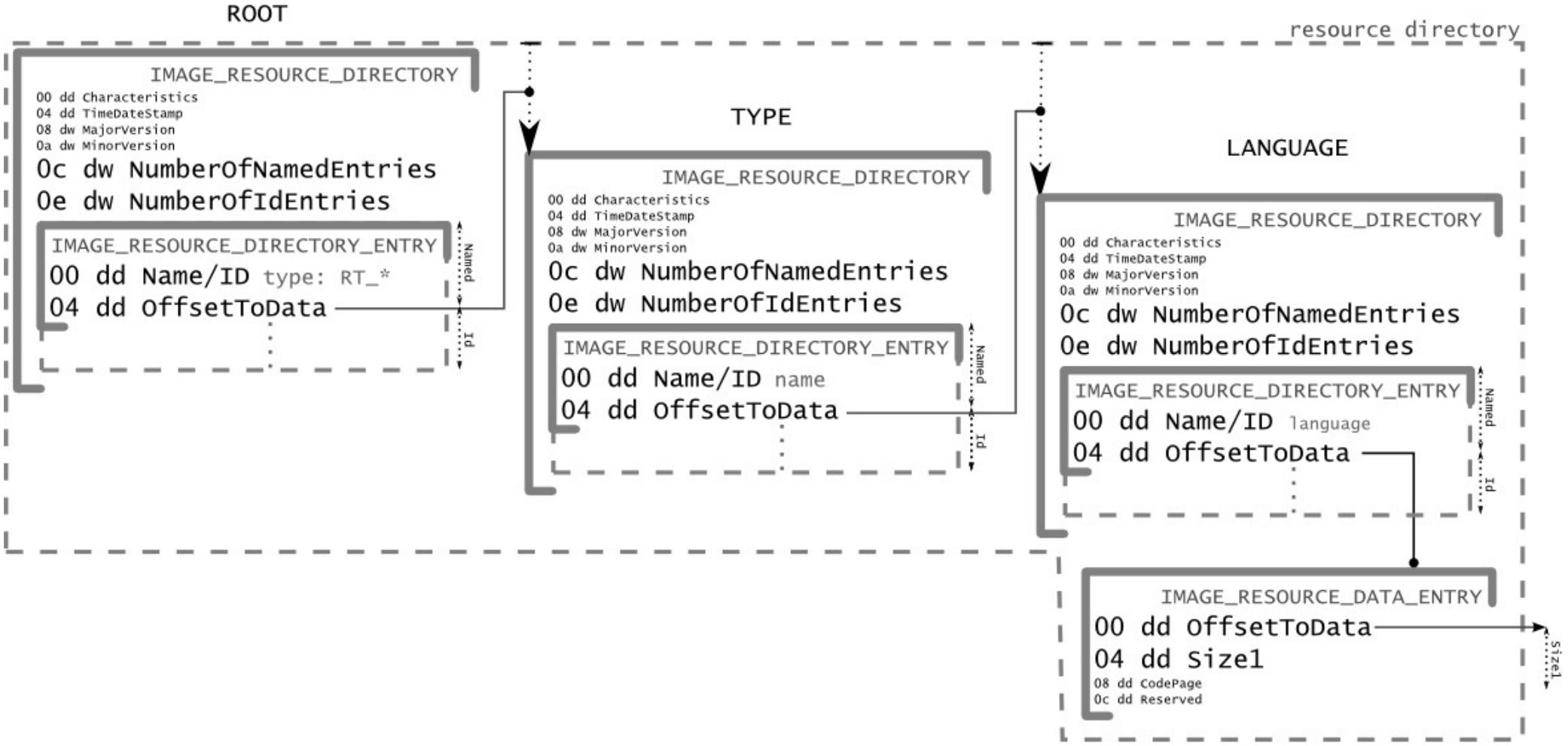
    → different ImageBase given by the loader
- *absolute* addresses need relocation
  - most addresses of the header are *relative*
  - immediate values in code, TLS callbacks
  - adds *(NewImageBase - OldImageBase)*


DIRECTORY.SIZE
SizeOfBlock
IMAGE_BASE_RELOCATION
00 dd VirtualAddress
04 dd SizeOfBlock
dw TypeOffset
relocation block
PUSH EBP
PUSH offset szMyString

Resources

- icons, dialogs, version information, ...
- requires only 3 APIs calls to be used

  → used everywhere
- folder & file structure
  - 3 levels in standard


ROOT
resource directory
IMAGE_RESOURCE_DIRECTORY
00 dd Characteristics
04 dd TimeDateStamp
08 dw MajorVersion
0a dw MinorVersion
0c dw NumberOfNamedEntries
0e dw NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00 dd Name/ID type: RT_*
04 dd OffsetToData
Named
Id
TYPE
IMAGE_RESOURCE_DIRECTORY
00 dd Characteristics
04 dd TimeDateStamp
08 dw MajorVersion
0a dw MinorVersion
0c dw NumberOfNamedEntries
0e dw NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00 dd Name/ID name
04 dd OffsetToData
Named
Id
LANGUAGE
IMAGE_RESOURCE_DIRECTORY
00 dd Characteristics
04 dd TimeDateStamp
08 dw MajorVersion
0a dw MinorVersion
0c dw NumberOfNamedEntries
0e dw NumberOfIdEntries
IMAGE_RESOURCE_DIRECTORY_ENTRY
00 dd Name/ID language
04 dd OffsetToData
Named
Id
IMAGE_RESOURCE_DATA_ENTRY
00 dd OffsetToData
04 dd Size1
08 dd CodePage
0c dd Reserved
Size1

*T*hread *L*ocal *S*torage

- Callbacks executed on thread start and stop
  - before EntryPoint
  - after ExitProcess


IMAGE_TLS_DIRECTORY
00 dd StartAddressOfRawData
04 dd EndAddressOfRawData
08 dd AddressOfIndex
pointer to index
0c dd AddressOfCallBacks
10 dd SizeOfZeroFill
14 dd Characteristics
00 dd Callback
dd 0

32 bits ↔ 64 bits

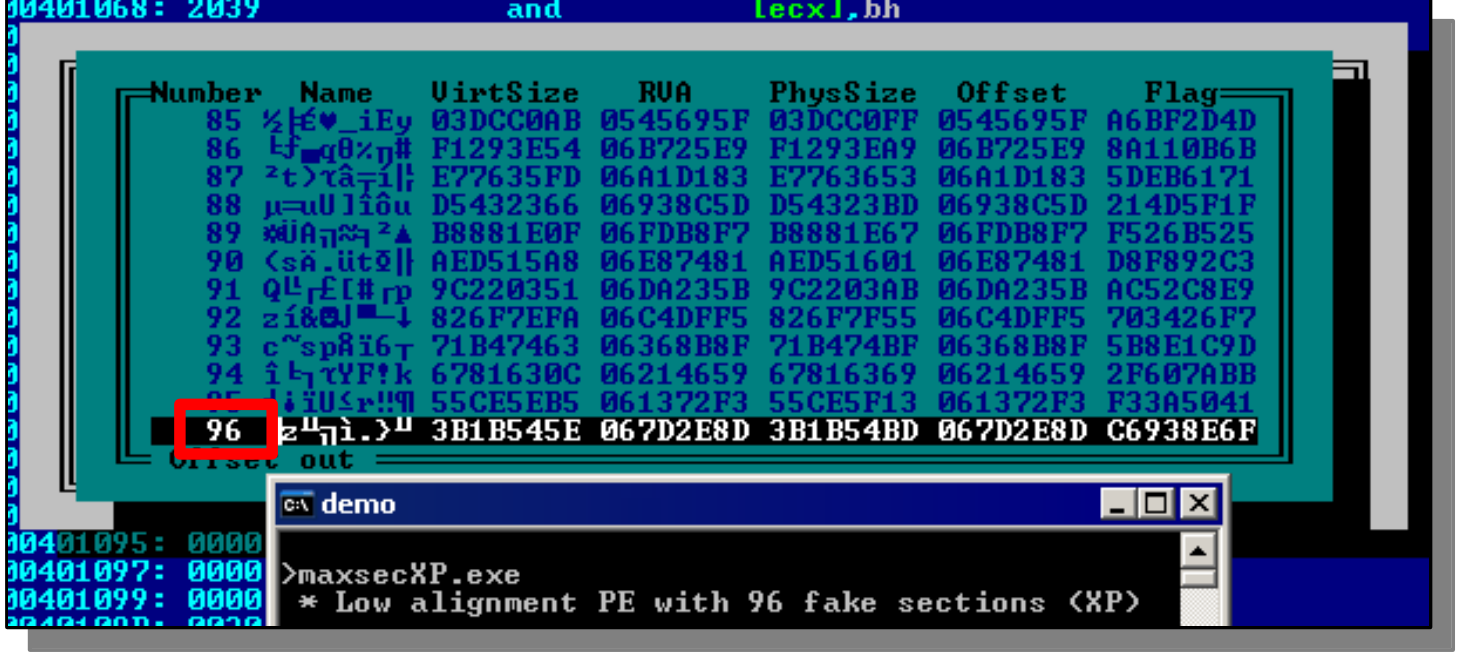
- IMAGE_FILE_HEADER.Machine
  - 0x14c *I386* ↔ 0x8664 *AMD64*
- IMAGE_OPTIONAL_HEADER.Magic
  - 0x10b ↔ 0x20b
- ImageBase, stack, heap
  - double ↔ quad
  - sizeof(OptionalHeader): 0xe0 ↔ 0xf0
- TLS, import thunks also switch to qwords

Classic tricks

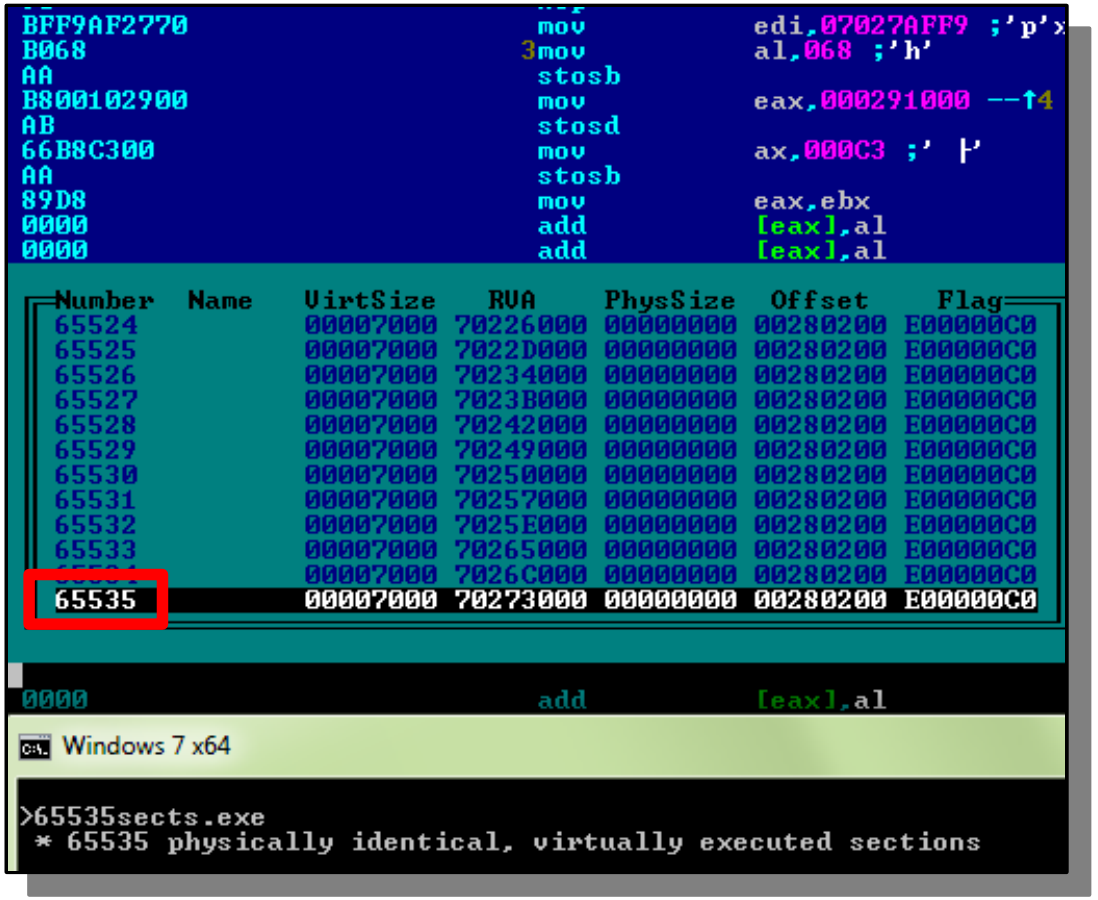
NumberOfSections

- 96 sections (XP)
- 65536 Sections (Vista or later)

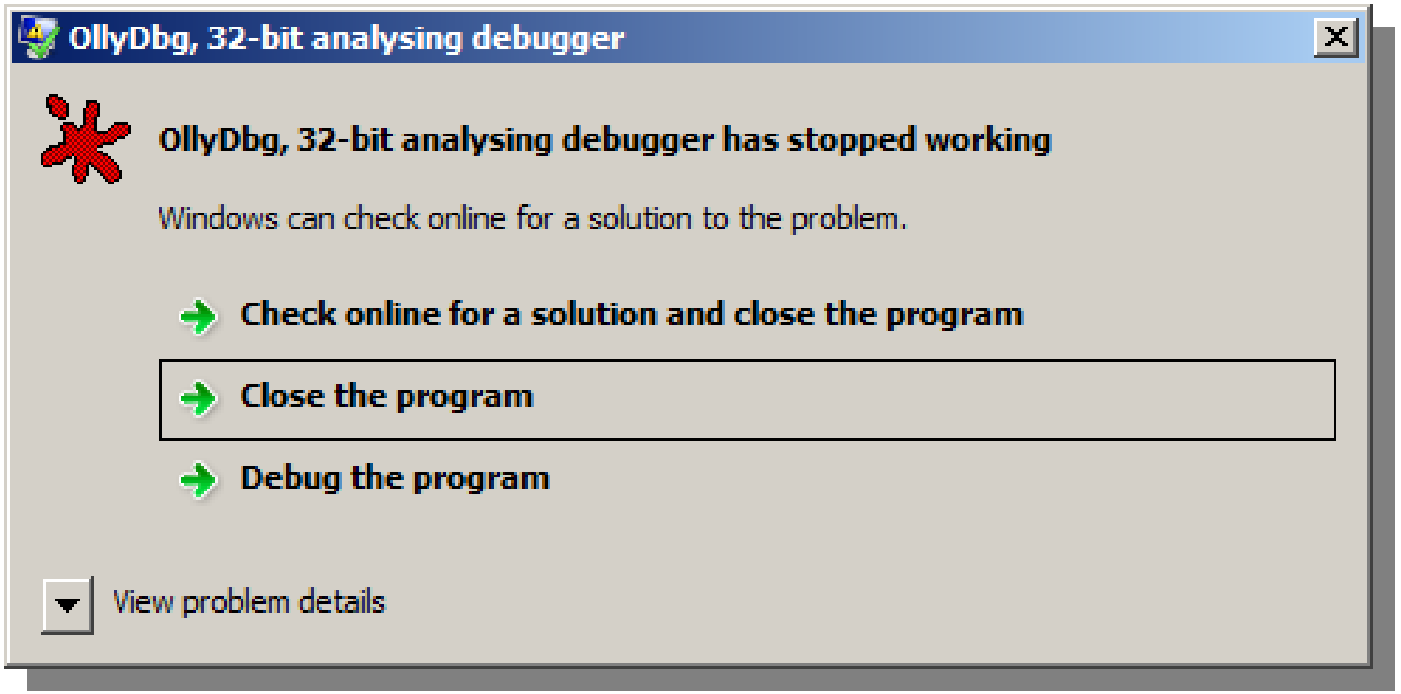
→ good enough to crash tools!



maxsecXP



65535sects


OllyDbg, 32-bit analysing debugger
OllyDbg, 32-bit analysing debugger has stopped working
Windows can check online for a solution to the problem.
Check online for a solution and close the program
Close the program
Debug the program
View problem details


Low memory!
Unable to allocate -531677184. bytes of memory
Don't display this message in the future
OK

SizeOfOptionalHeader

- ~~sizeof(OptionalHeader)~~
  - that would be 0xe0 (32b)/0xf0 (64b)
  - many naive softwares fail if different
- offset(SectionTable) – offset(OptionalHeader)
- can be:
  - bigger
    - bigger than file (→ virtual table, xp)
  - smaller or null (→ overlapping OptionalHeader)
  - null (no section at all)

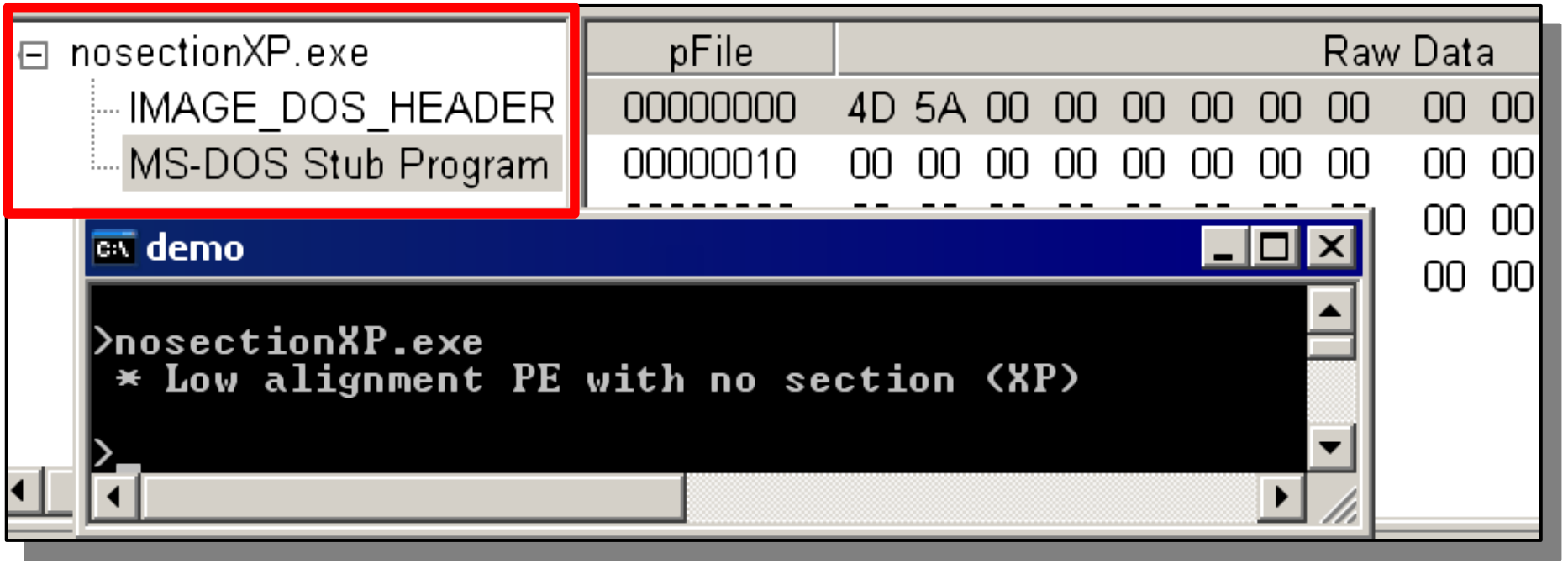
Section-less PE

- standard mode:
  - 200 ≤ FileAlignment ≤ SectionAlignment
  - 1000 ≤ SectionAlignment
- 'drivers' mode:
  - 1 ≤ FileAlignment == SectionAlignment ≤ 800

→ virtual == physical

- whole file mapped as is
- sections are meaningless
  - can be none, can be many (bogus or not)

1 ≤ FileAlignment == SectionAlignment ≤ 800



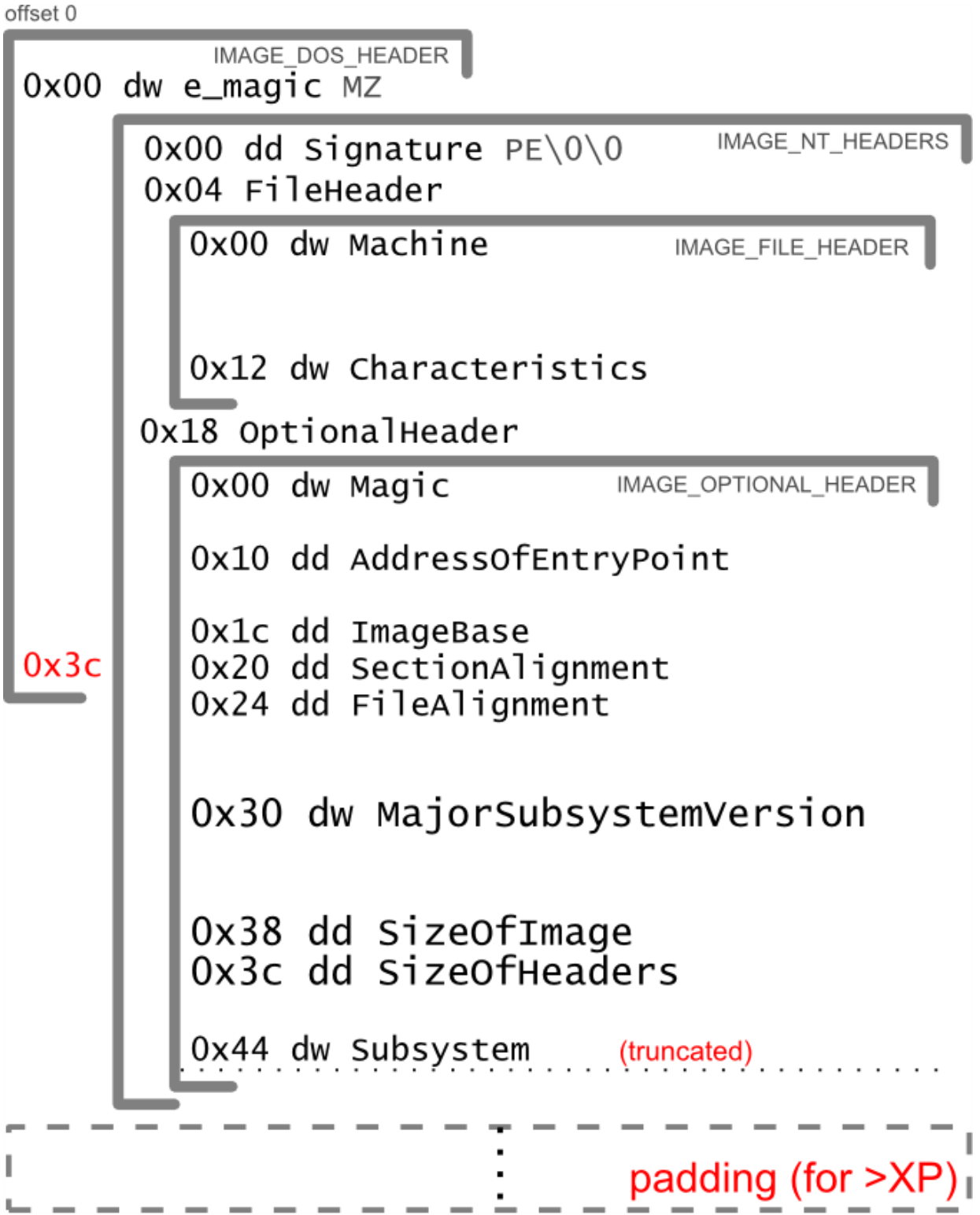
TinyPE

classic example of hand-made malformation

- PE header in Dos header
- truncated OptionalHeader

- doesn't require a section
- 64b & driver compatible
- 92 bytes
  - XP only (no more truncated OptionalHeader)
  - extra padding is required since Vista

  → smallest universal PE: 268 bytes


offset 0
IMAGE_DOS_HEADER
0x00 dw e_magic MZ
0x00 dd Signature PE\0\0
IMAGE_NT_HEADERS
0x04 FileHeader
0x00 dw Machine
IMAGE_FILE_HEADER
0x12 dw Characteristics
0x18 OptionalHeader
0x00 dw Magic
IMAGE_OPTIONAL_HEADER
0x10 dd AddressOfEntryPoint
0x1c dd ImageBase
0x20 dd SectionAlignment
0x24 dd FileAlignment
0x3c
0x30 dw MajorSubsystemVersion
0x38 dd SizeOfImage
0x3c dd SizeOfHeaders
0x44 dw Subsystem
(truncated)
padding (for >XP)

tiny*

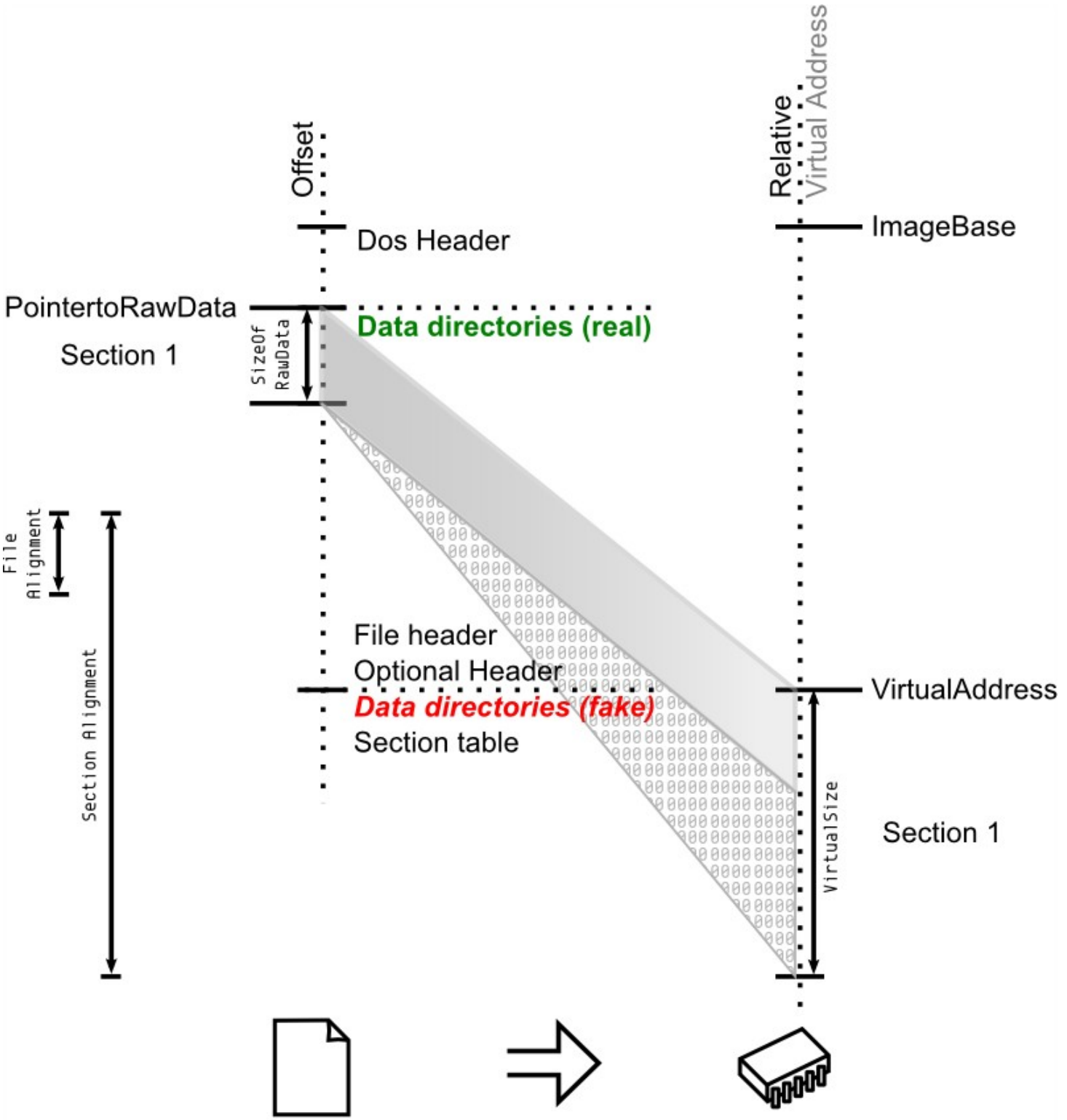
Dual 'folded' headers

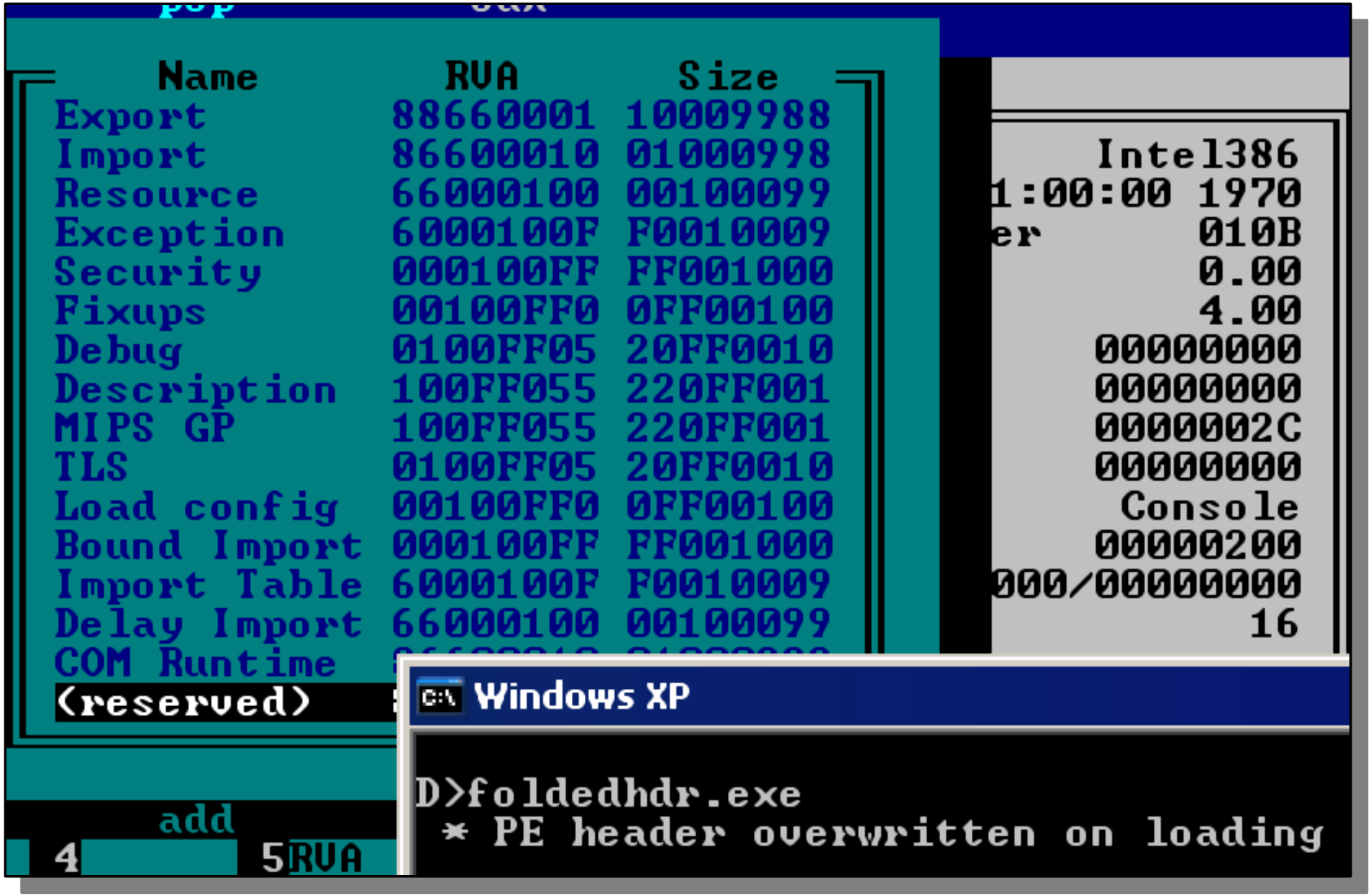
DD only used **after** mapping
http://www.reversinglabs.com/advisory/pecoff.php

1. move down header
2. fake DD overlaps starts of section (hex art FTW)
3. section area contains real values

- loading process:

1. header and sections are parsed
2. file is mapped
3. DD overwritten with real value
    - imports are resolved, etc...


Offset
Relative Virtual Address
Dos Header
ImageBase
PointertoRawData
Section 1
SizeOf RawData
Data directories (real)
File Alignment
Section Alignment
File header
Optional Header
Data directories (fake)
Section table
VirtualAddress
VirtualSize
Section 1

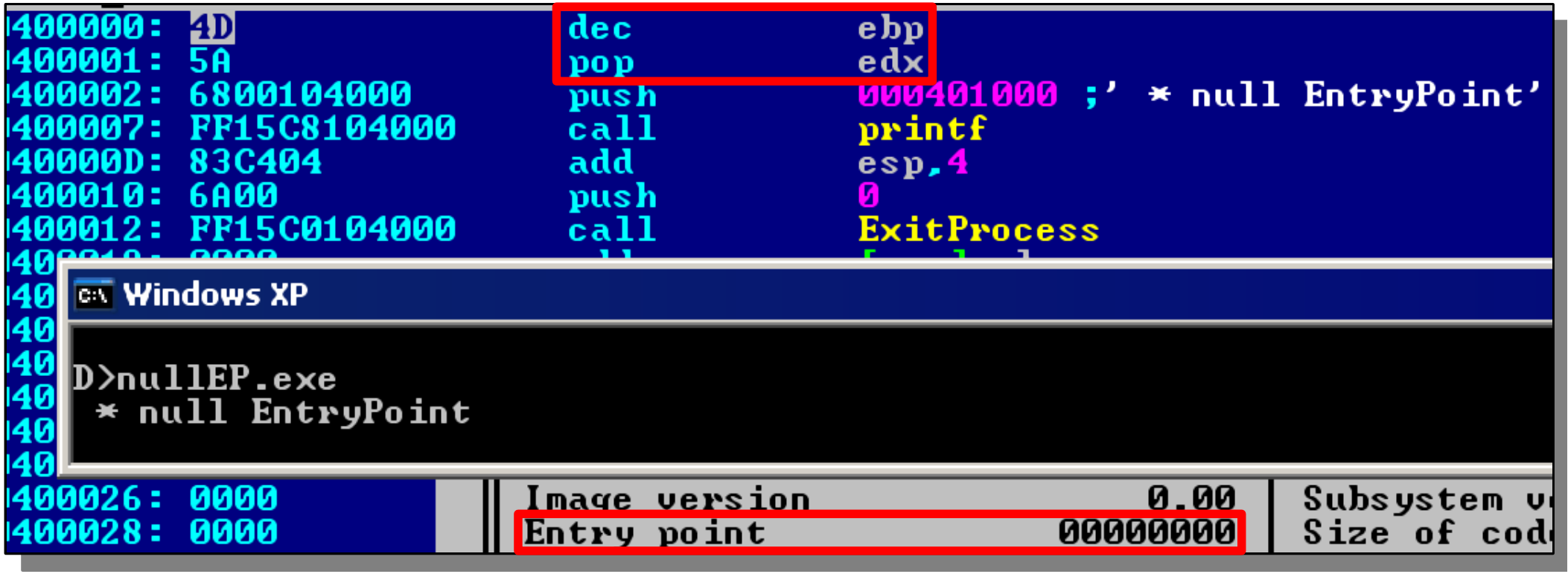

Name RVA Size
Export 88660001 10009988
Import 86600010 01000998
Resource 66000100 00100099
Exception 6000100F F0010009
Security 000100FF FF001000
Fixups 00100FF0 0FF00100
Debug 0100FF05 20FF0010
Description 100FF055 220FF001
MIPS GP 100FF055 220FF001
TLS 0100FF05 20FF0010
Load config 00100FF0 0FF00100
Bound Import 000100FF FF001000
Import Table 6000100F F0010009
Delay Import 66000100 00100099
COM Runtime
(reserved)
Intel386
1:00:00 1970
er 010B
0.00
4.00
00000000
00000000
0000002C
00000000
Console
00000200
000/00000000
16
Windows XP
D>foldedhdr.exe
* PE header overwritten on loading
add
4
5RVA

foldedhdr

null EntryPoint

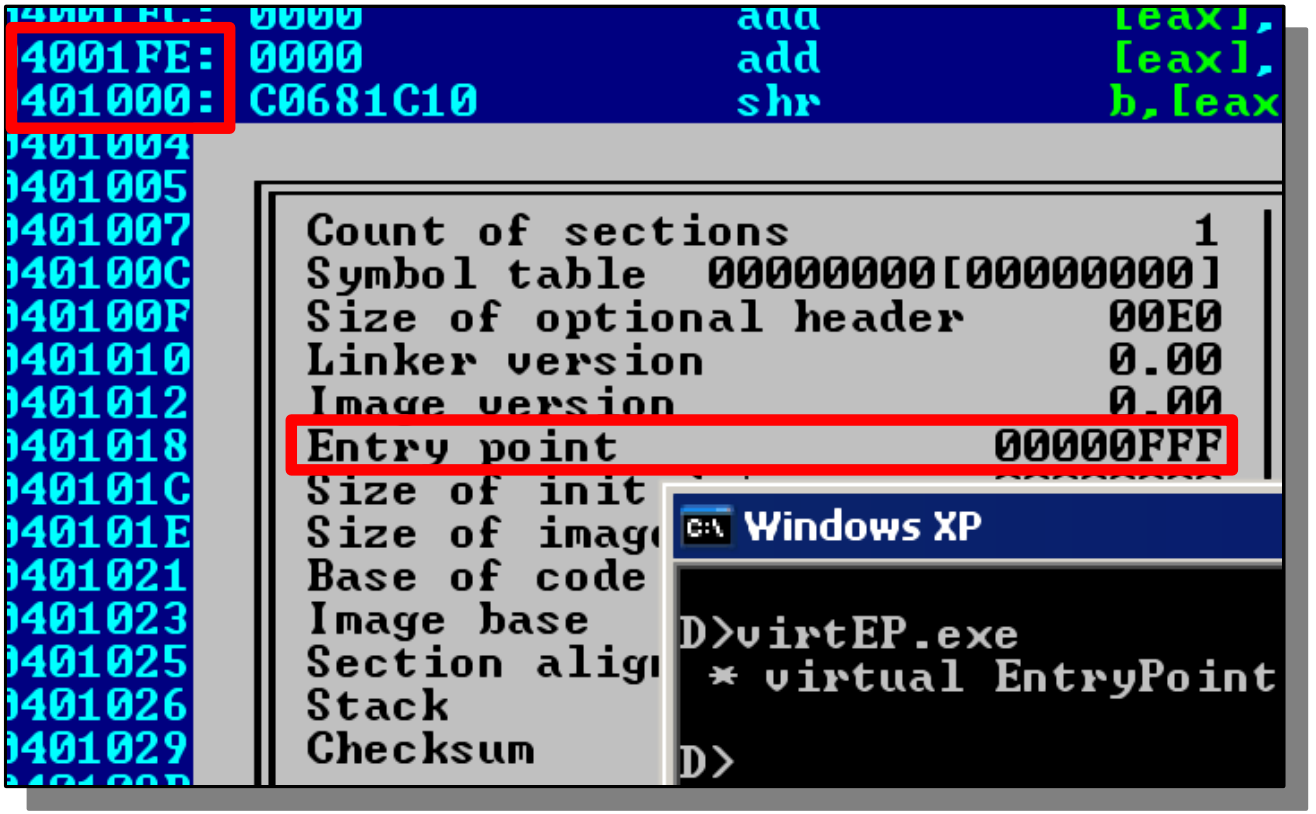
- for EXEs
  - 'MZ' disassembled as 'dec ebp/pop edx'

    (null EP for DLLs = no DllMain call)



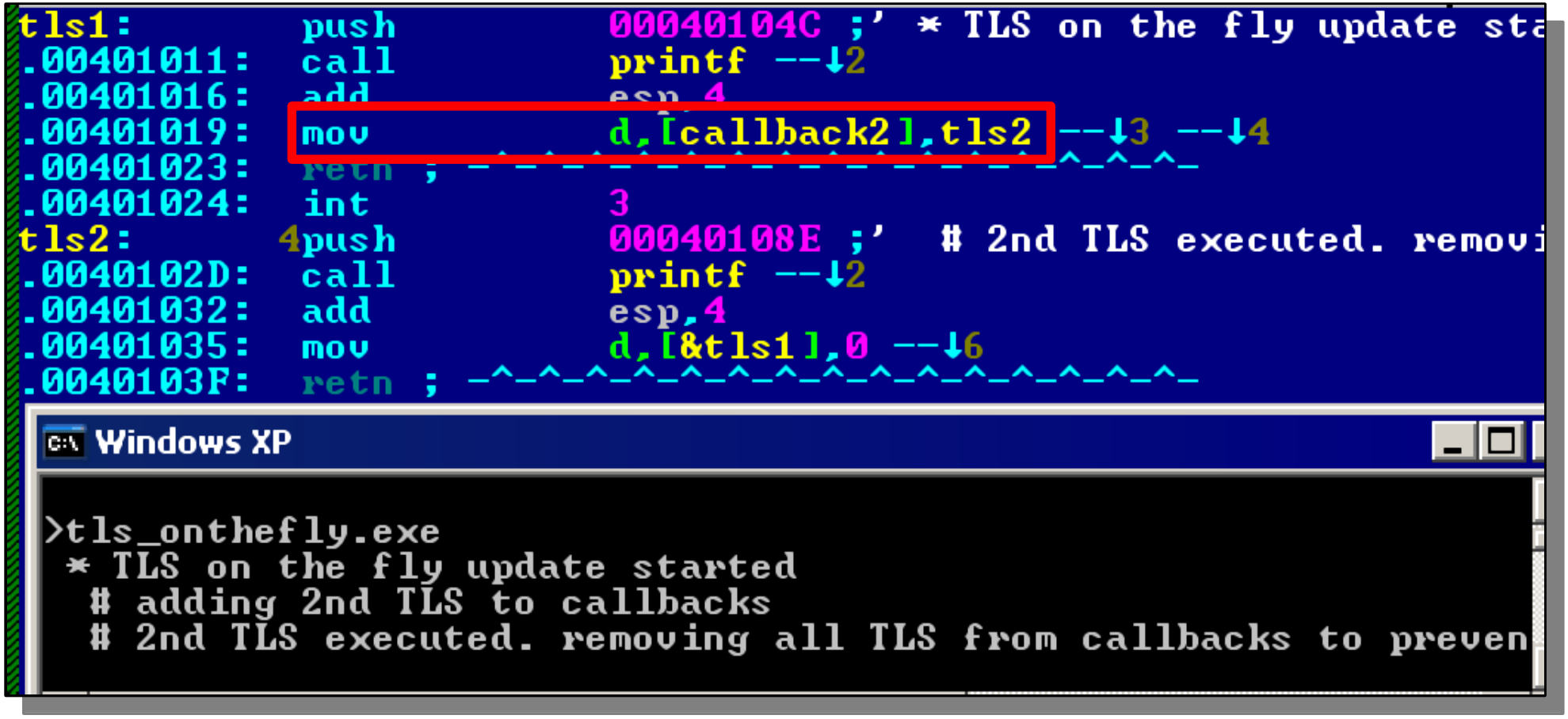
virtual EntryPoint

- first byte not physically in the file
  - 00 C0 => add al, al



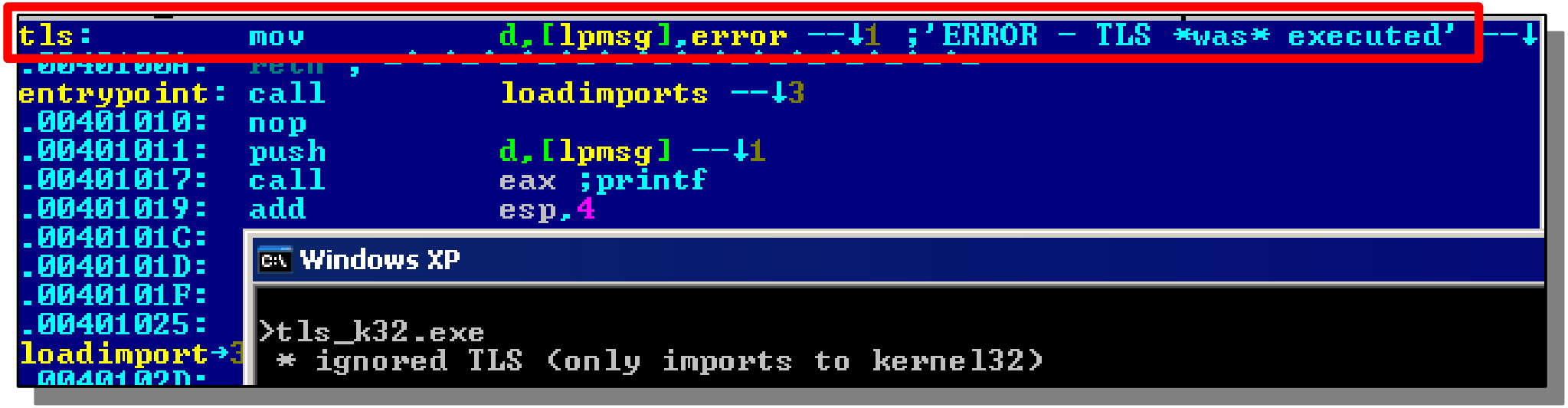
TLS on the fly

- the list of callbacks is updated on the fly
  - add callback #2 during callback #1



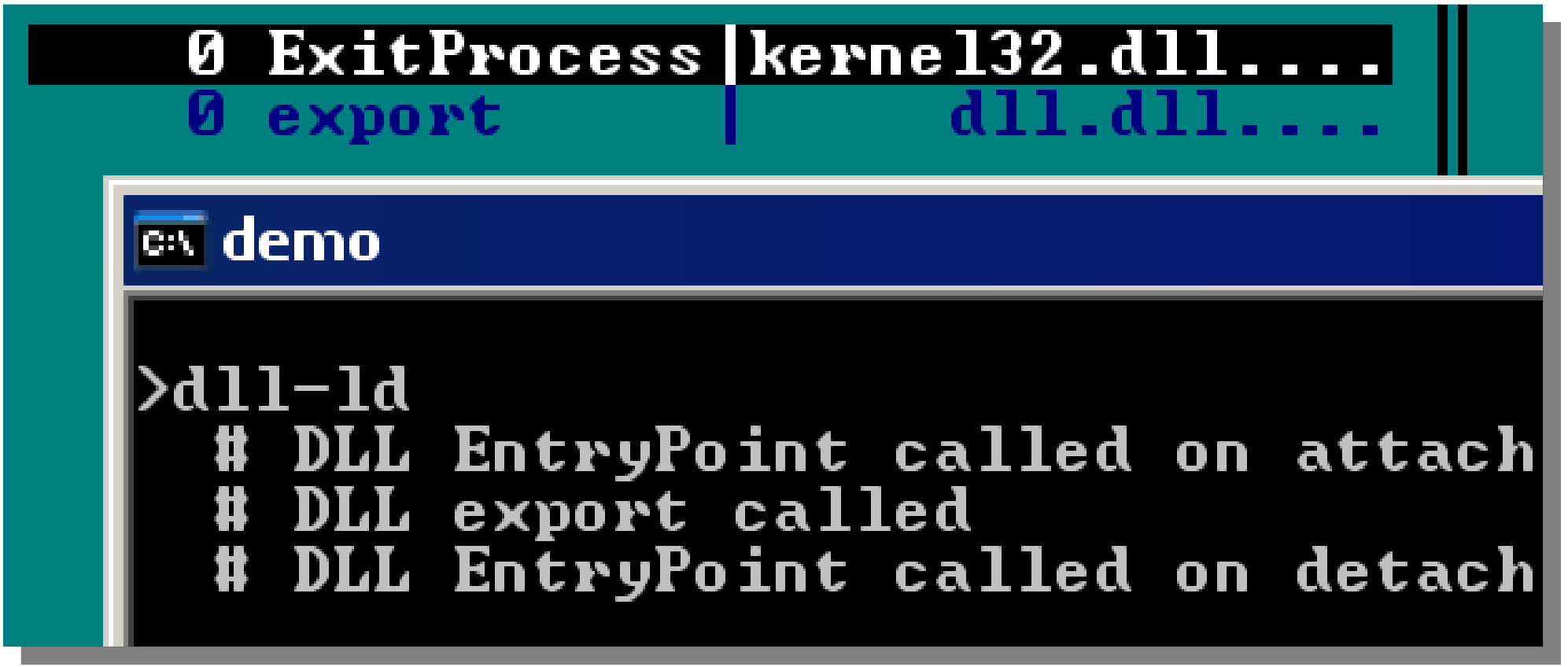
ignored TLS

- TLS are **not** executed if only kernel32 is imported
  - and if no DLL importing kernel32 is imported
    - Kaspersky & Ferrie



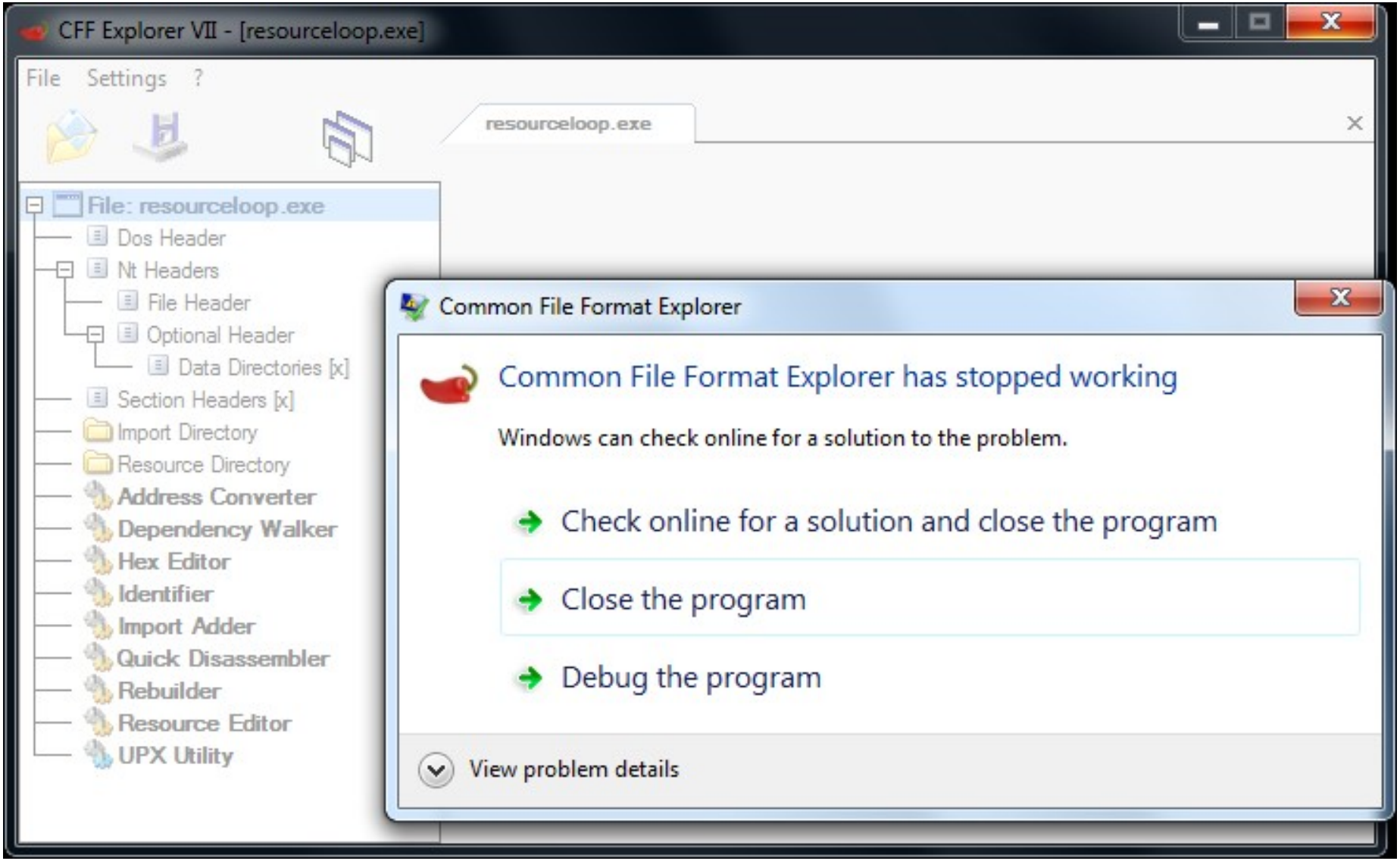
imports' trailing dots

- XP only
- trivial
  - trailing dots are ignored after DLL name
- fails heuristics


0 ExitProcess|kernel32.dll....
0 export | dll.dll....
demo
>dll-ld
DLL EntryPoint called on attach
DLL export called
DLL EntryPoint called on detach

Resources loops

- (infinite) loops
  - not checked by the loader
  - ignored if a different path is required to reach resource



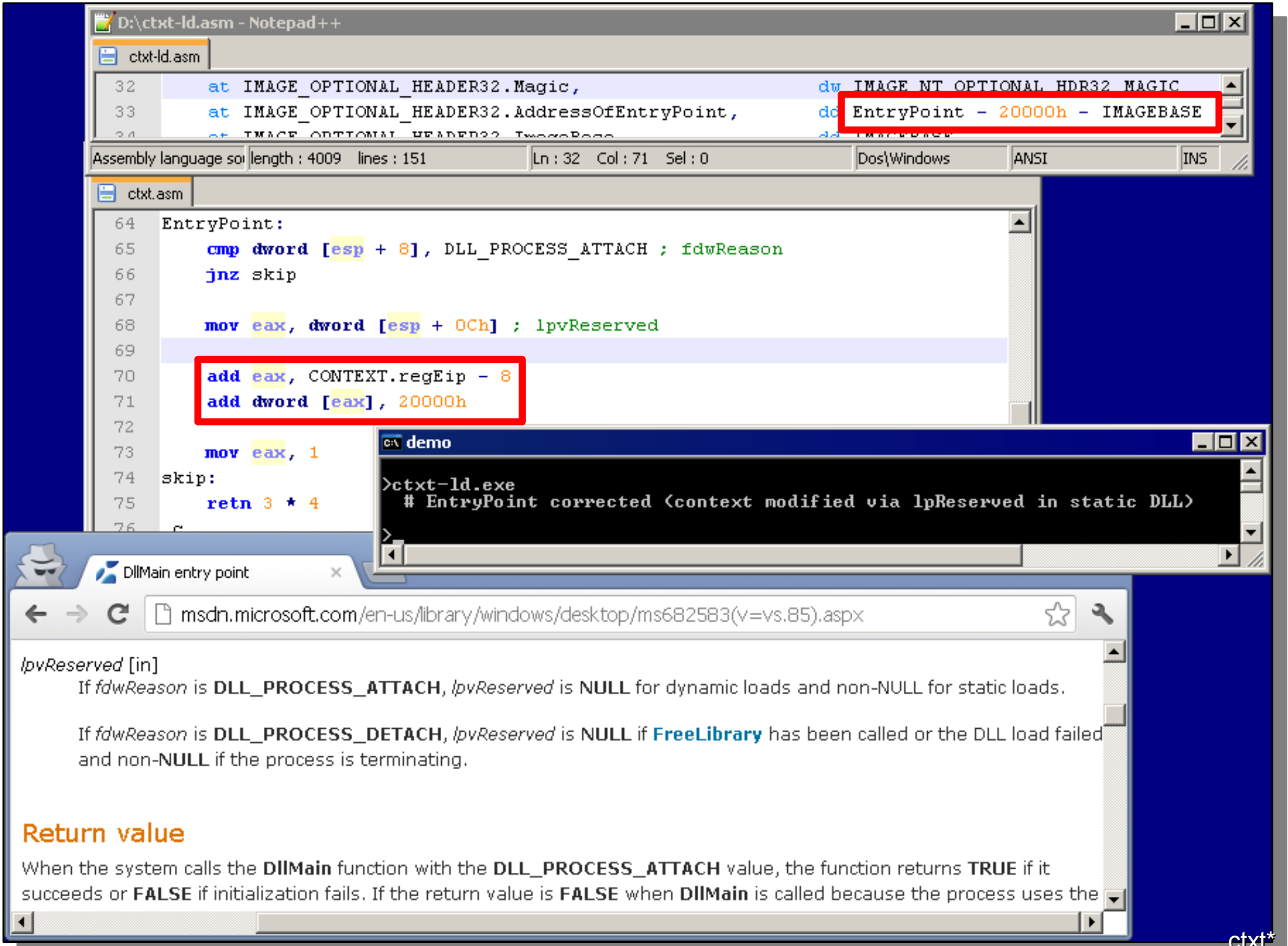
EntryPoint change via static DLLs

static DLLs are called before EntryPoint call

- DllMain gets thread context via *lpvReserved*
  - which already contains the future EntryPoint

→ any static DLL can freely change the EntryPoint

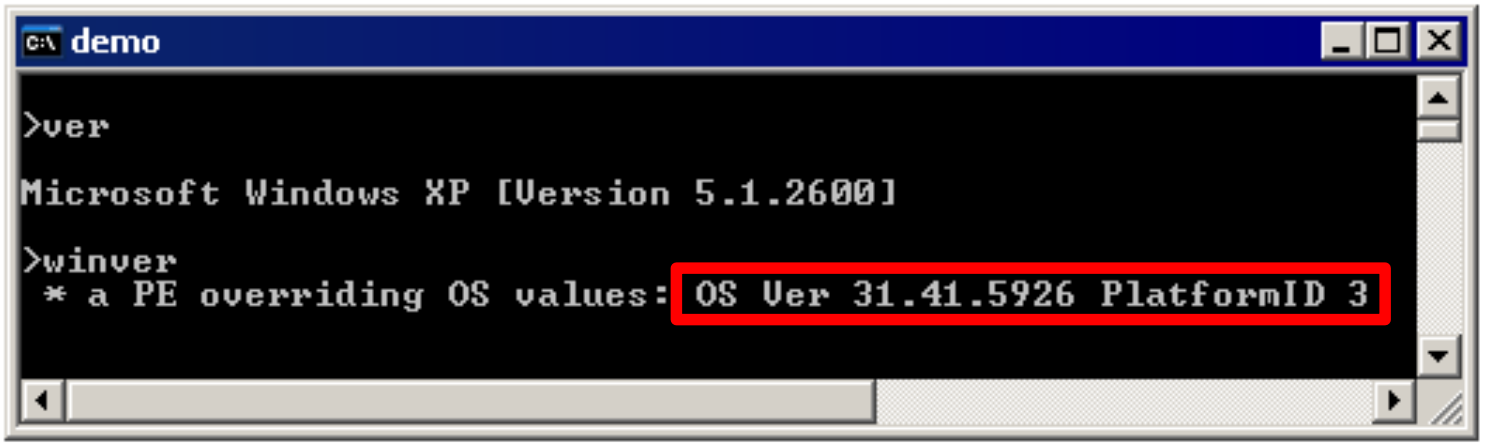
documented by Skywing (http://www.nynaeve.net/?p=127),
but not widely known


D:\ctxt-ld.asm - Notepad++
ctxt-ld.asm
32 at IMAGE_OPTIONAL_HEADER32.Magic, dw IMAGE_NT_OPTIONAL_HDR32_MAGIC
33 at IMAGE_OPTIONAL_HEADER32.AddressOfEntryPoint, dd EntryPoint - 20000h - IMAGEBASE
Assembly language so length : 4009 lines : 151 Ln : 32 Col : 71 Sel : 0 Dos\Windows ANSI INS
ctxt.asm
64 EntryPoint:
65 cmp dword [esp + 8], DLL_PROCESS_ATTACH ; fdwReason
66 jnz skip
67
68 mov eax, dword [esp + 0Ch] ; lpvReserved
69
70 add eax, CONTEXT.regEip - 8
71 add dword [eax], 20000h
72
73 mov eax, 1
74 skip:
75 retn 3 * 4
demo
>ctxt-ld.exe
EntryPoint corrected (context modified via lpReserved in static DLL)
>
DllMain entry point
msdn.microsoft.com/en-us/library/windows/desktop/ms682583(v=vs.85).aspx
lpvReserved [in]
If fdwReason is DLL_PROCESS_ATTACH, lpvReserved is NULL for dynamic loads and non-NULL for static loads.
If fdwReason is DLL_PROCESS_DETACH, lpvReserved is NULL if FreeLibrary has been called or the DLL load failed and non-NULL if the process is terminating.
Return value
When the system calls the DllMain function with the DLL_PROCESS_ATTACH value, the function returns TRUE if it succeeds or FALSE if initialization fails. If the return value is FALSE when DllMain is called because the process uses the
ctxt*

Win32VersionValue

- officially *reserved*
  - 'should be null'
- actually used to override versions info in the PEB
- simple dynamic anti-emu
  - used in malwares

```
[...]
OSMAJOR equ 31
OSMINOR equ 41
BUILD equ 5926
ID equ 3 ; [0;3]

[...]

at Win32VersionValue,  dd OSMAJOR | (OSMINOR << 8) | ((BUILD & 03fffh) << 16) | (((ID & 3) ^ 02h) << 30)

[...]

EntryPoint:
    push OSVerEx
    call [__imp__GetVersionExA]
[...]
```

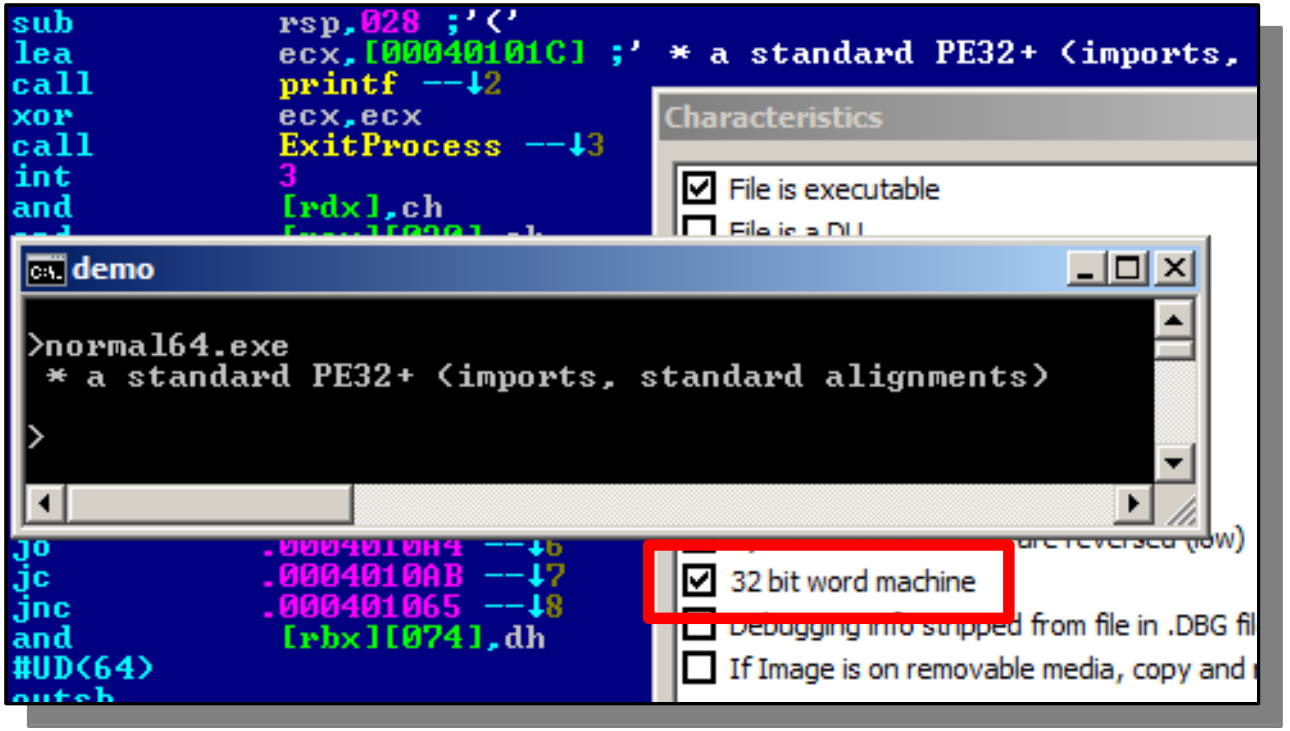


winver

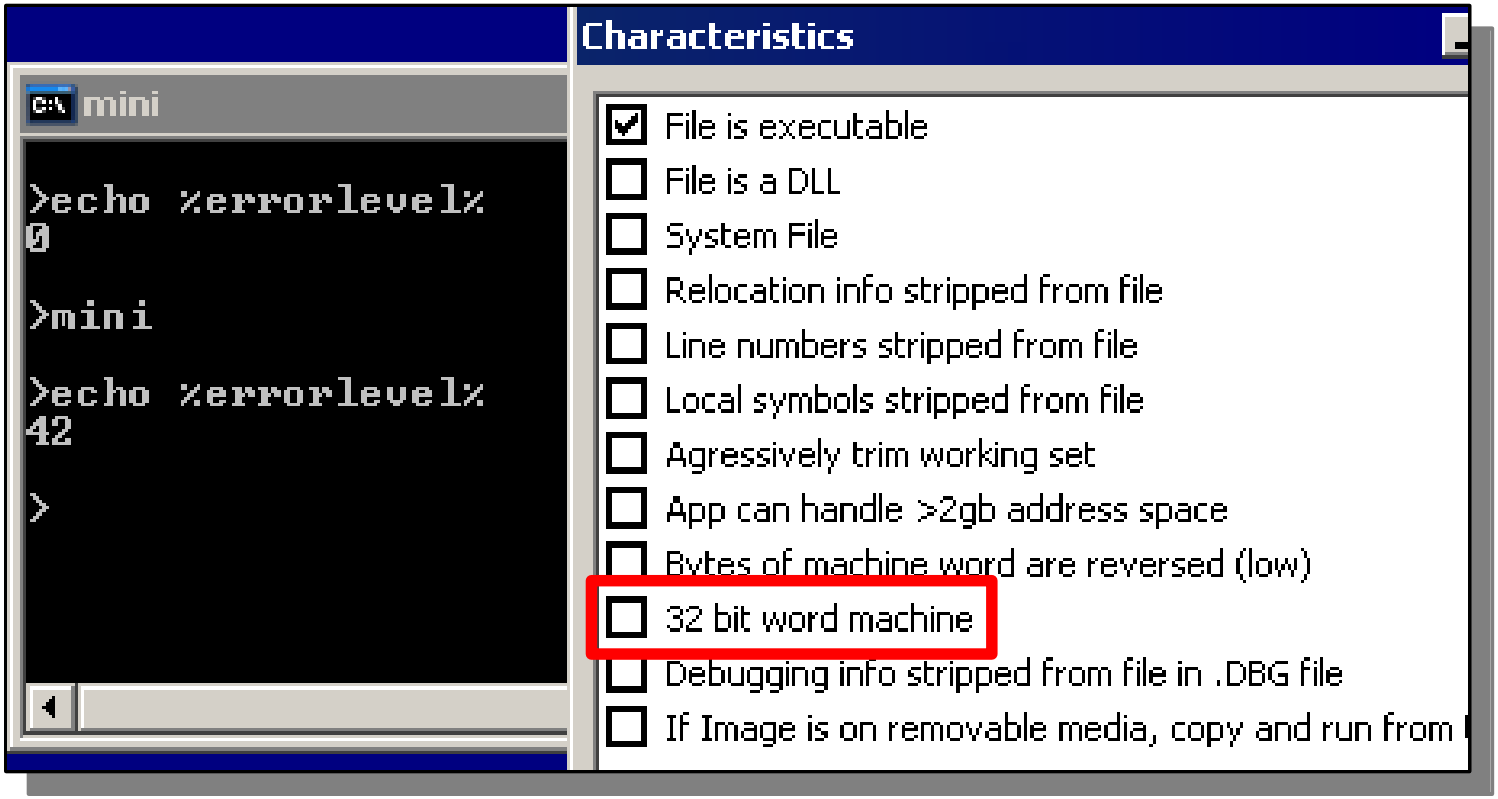
tricks

Characteristics

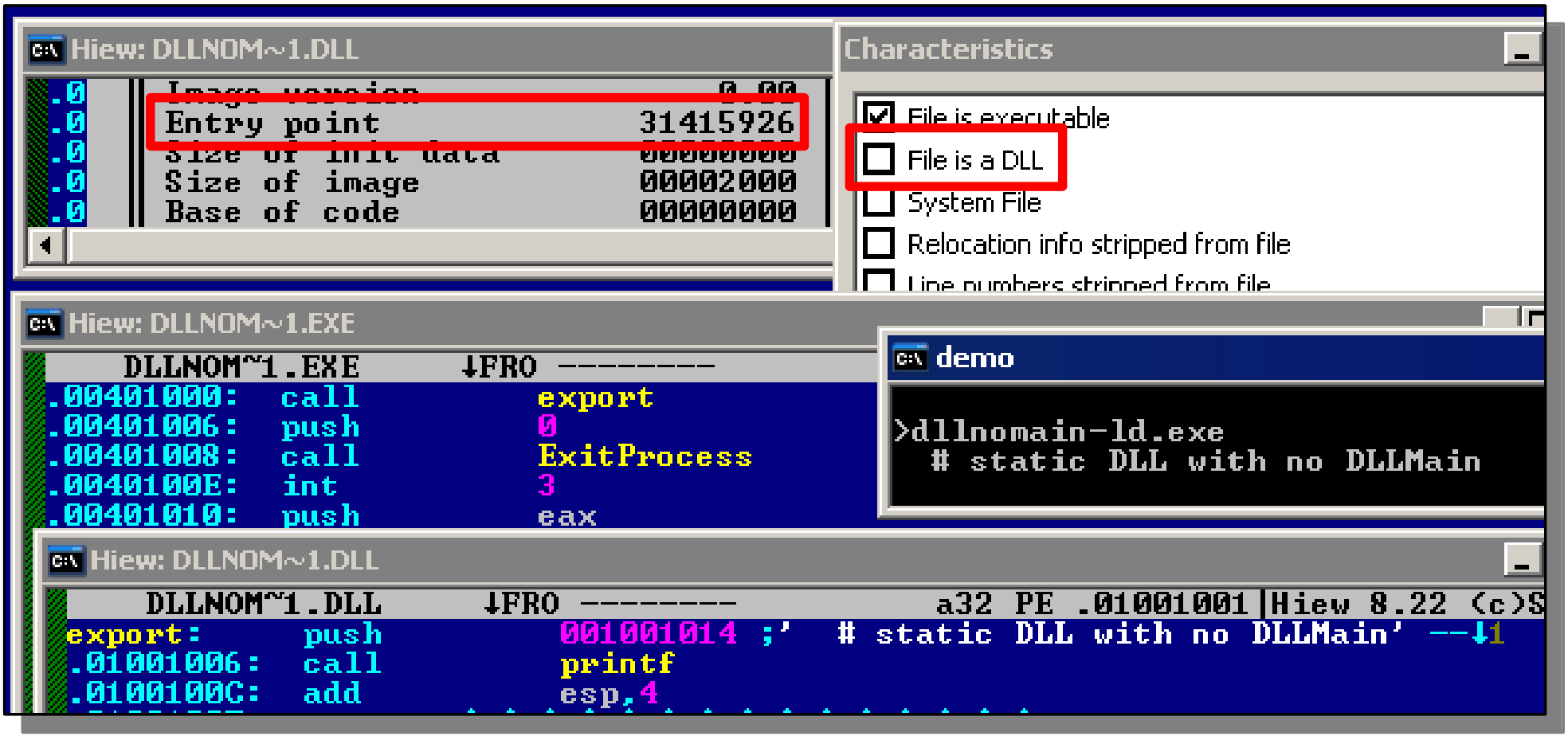
- IMAGE_FILE_32BIT_MACHINE
  - true for 64b
  - ***not*** required !!
- IMAGE_FILE_DLL
  - ***not*** required in DLLs
    - exports still useable
    - no DllMain call!
      - invalid EP → not an EXE
      - no FILE_DLL → apparently not a DLL
        → can't be debugged



normal64



mini

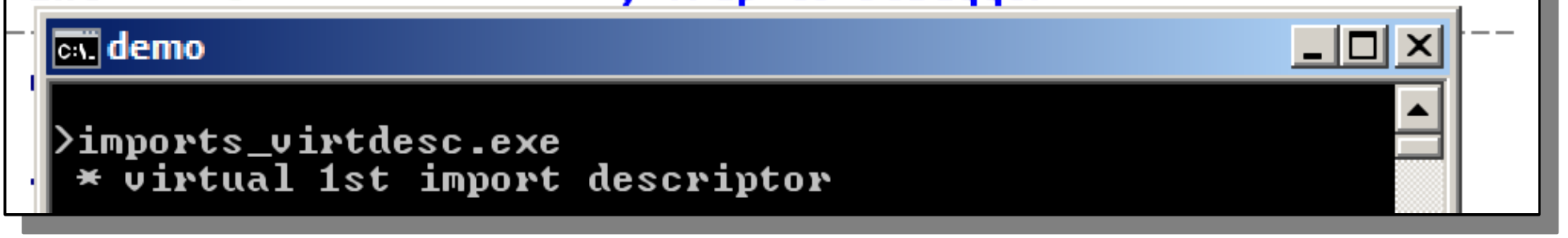

Hiew: DLLNOM~1.DLL
Entry point 31415926
Size of image 00002000
Base of code 00000000
Characteristics
File is executable
File is a DLL
System File
Relocation info stripped from file
Hiew: DLLNOM~1.EXE
DLLNOM~1.EXE
.00401000: call export
.00401006: push 0
.00401008: call ExitProcess
.0040100E: int 3
.00401010: push eax
demo
>dllnomain-ld.exe
static DLL with no DLLMain
Hiew: DLLNOM~1.DLL
DLLNOM~1.DLL
a32 PE .01001001
export: push 01001014 ;' # static DLL with no DLLMain'
.01001006: call printf
.0100100C: add esp,4

Imports descriptor tricks

- INT bogus or absent
  - only DllName and IAT required
- descriptor just skipped if no thunk
  - DLL name ignored
    - can be null or VERY big
  - parsing shouldn't abort too early
- isTerminator = (IAT == 0 || DllName == 0)
- terminator can be virtual or outside file
  - first descriptor too

dd OriginalFirstThunk
dd TimeDateStamp
dd ForwarderChain
---
dd Name
dd FirstThunk

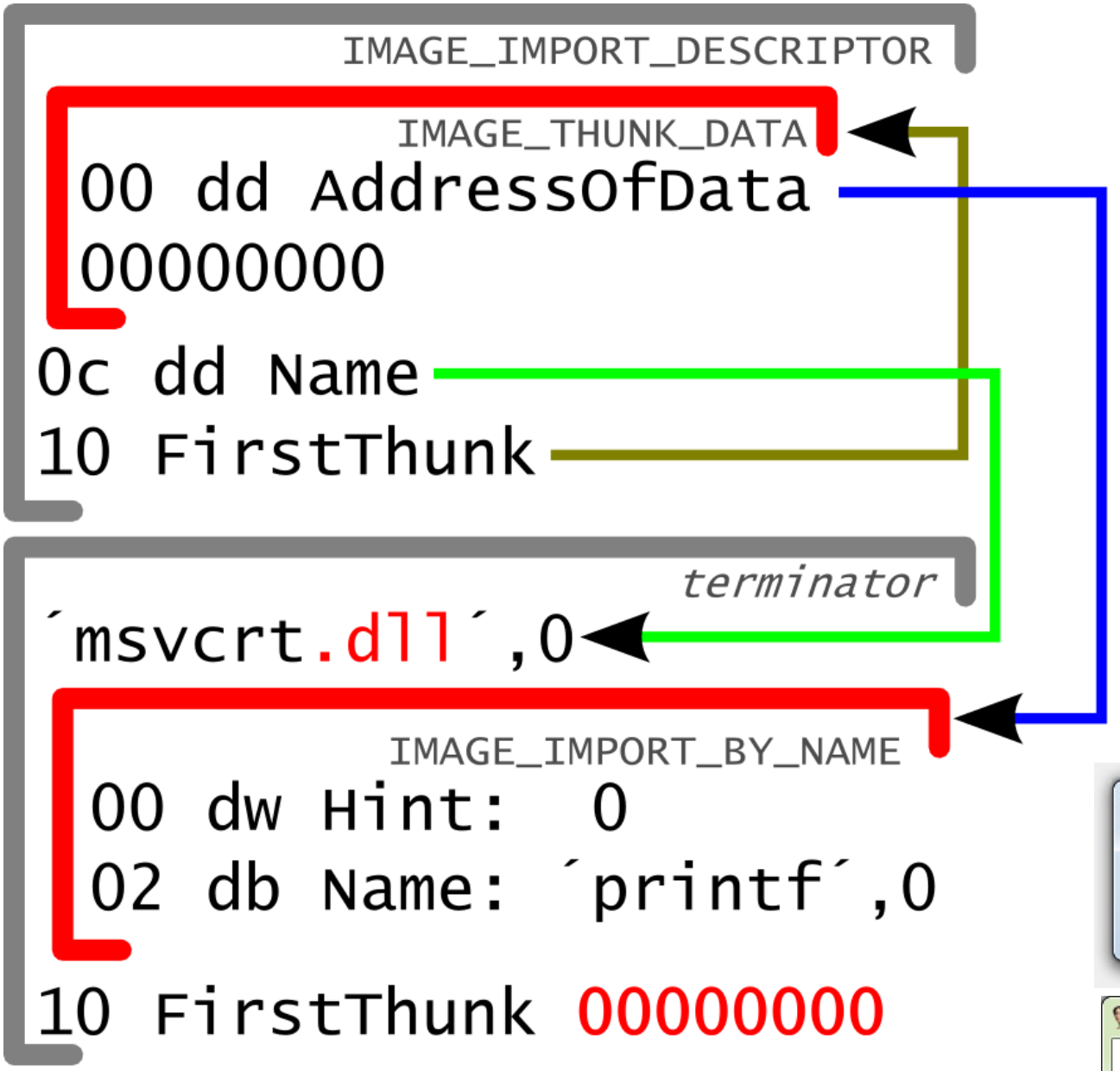
```
                                ; " * virtual 1st import descriptor\n"
push    offset aVirtual1stImpo
call    dword ptr byte_401081+7
add     esp, 4
nop
push    0
call    dword ptr sub_401080
int     3                       ; Trap to Debugger
```

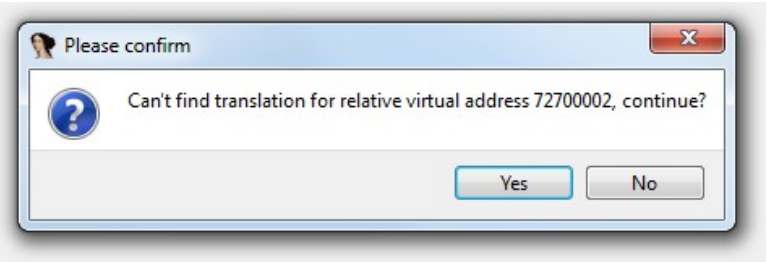


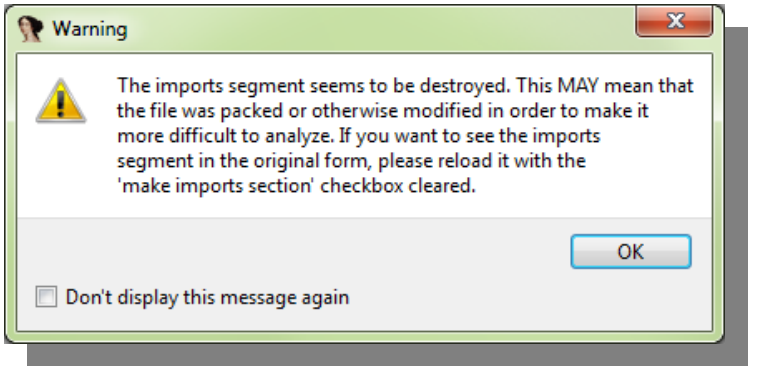
Collapsed imports

advanced imports malformation

- extension-less DLL name
- IAT in descriptor
  - pseudo-valid INT that is ignored
- name and hint/names in terminator
- valid because last dword is null


IMAGE_IMPORT_DESCRIPTOR
IMAGE_THUNK_DATA
00 dd AddressOfData
00000000
0c dd Name
10 FirstThunk
terminator
´msvcrt.dll´,0
IMAGE_IMPORT_BY_NAME
00 dw Hint: 0
02 db Name: ´printf´,0
10 FirstThunk 00000000


Please confirm
Can't find translation for relative virtual address 72700002, continue?
Yes
No

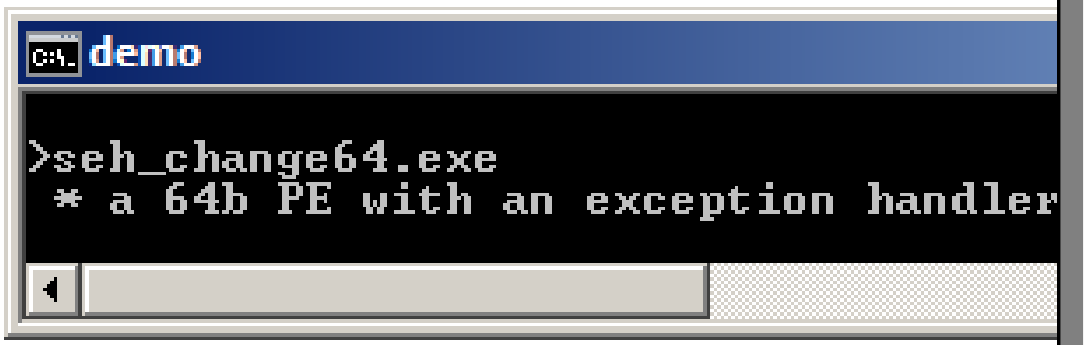

Warning
The imports segment seems to be destroyed. This MAY mean that the file was packed or otherwise modified in order to make it more difficult to analyze. If you want to see the imports segment in the original form, please reload it with the 'make imports section' checkbox cleared.
OK
Don't display this message again

Exceptions directory

- 64 bits **S**tructured **E**xception **H**andler
  - usually with a lot of extra compiler code
- used by W32.Deelae for infection
  - Peter Ferrie, Virus Bulletin September 2011
- update-able manually, on the fly
  - no need to go through APIs

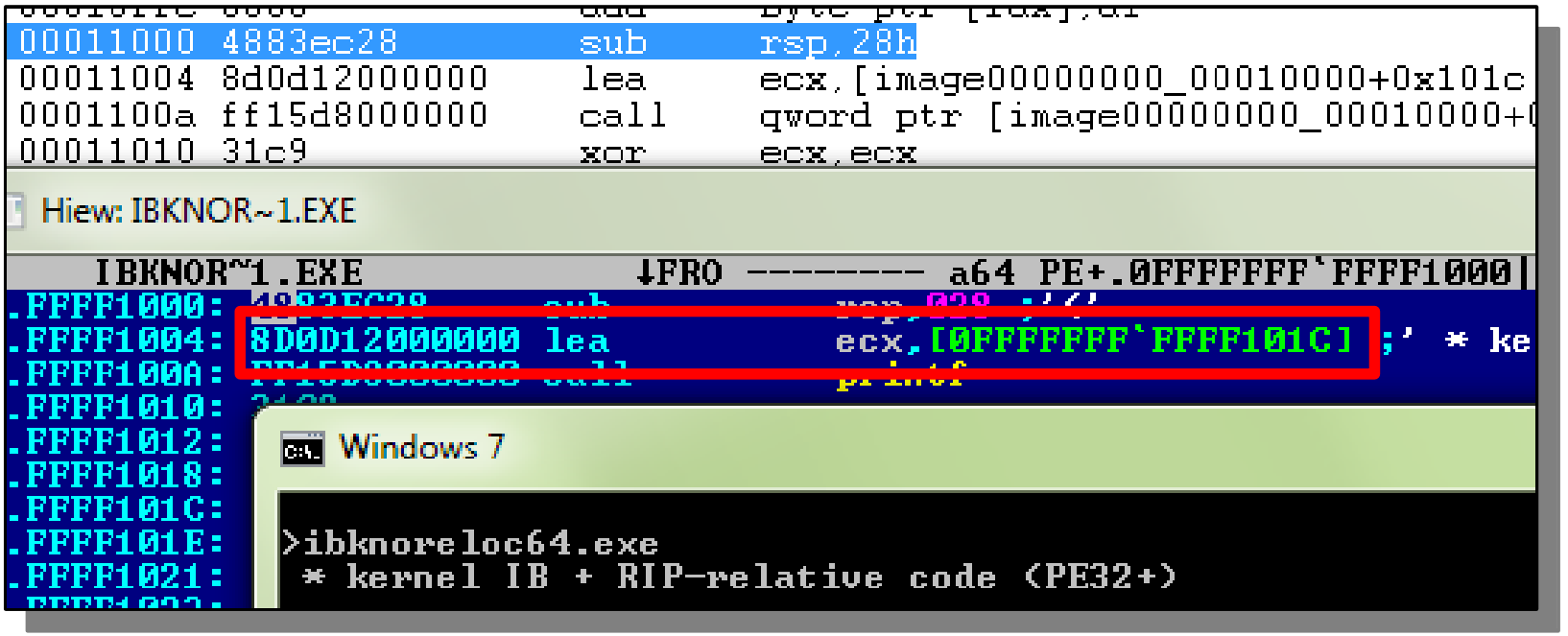
```
...
    at IMAGE_DATA_DIRECTORY_16.Exception, dd Exception - IMAGEBASE, EXCEPTION_SIZE
...

Exception:
istruc RUNTIME_FUNCTION
    at RUNTIME_FUNCTION.FunctionStart, dd EntryPoint - IMAGEBASE
    at RUNTIME_FUNCTION.FunctionEnd  , dd EndCode - IMAGEBASE
    at RUNTIME_FUNCTION.UnwindInfo   , dd UnwindData - IMAGEBASE
iend
EXCEPTION_SIZE equ $ - Exception

UnwindData:
istruc UNWIND_INFO
    at UNWIND_INFO.Ver3_Flags     , db 1 + (UNW_FLAG_EHANDLER << 3)
    at UNWIND_INFO.CntUnwindCodes , db 0 ; let's shrink it to the minimum
iend
align 4, db 0
    dd Handler - IMAGEBASE ; 1 exception handler
    dd 0                   ; handler data
...

EntryPoint:
    sub rsp, 8 * 5
    int3
EndCode:

...

Handler:
    lea ecx, [Msg]
    call [__imp__printf]
...

Msg db " * a 64b PE making use of an Exception DataDirectory", 0ah, 0
...
```

```
...
EntryPoint:
    sub rsp, 8 * 5
    add dword [HandlerRVA], NewHandler - InitialHandler
    int3
...
HandlerRVA:
    dd InitialHandler - IMAGEBASE
...

NewHandler:
    lea ecx, [NewMsg]
    call [__imp__printf]
...
NewMsg db " * a 64b PE with an exception handler address modified on the fly", 0
...
```



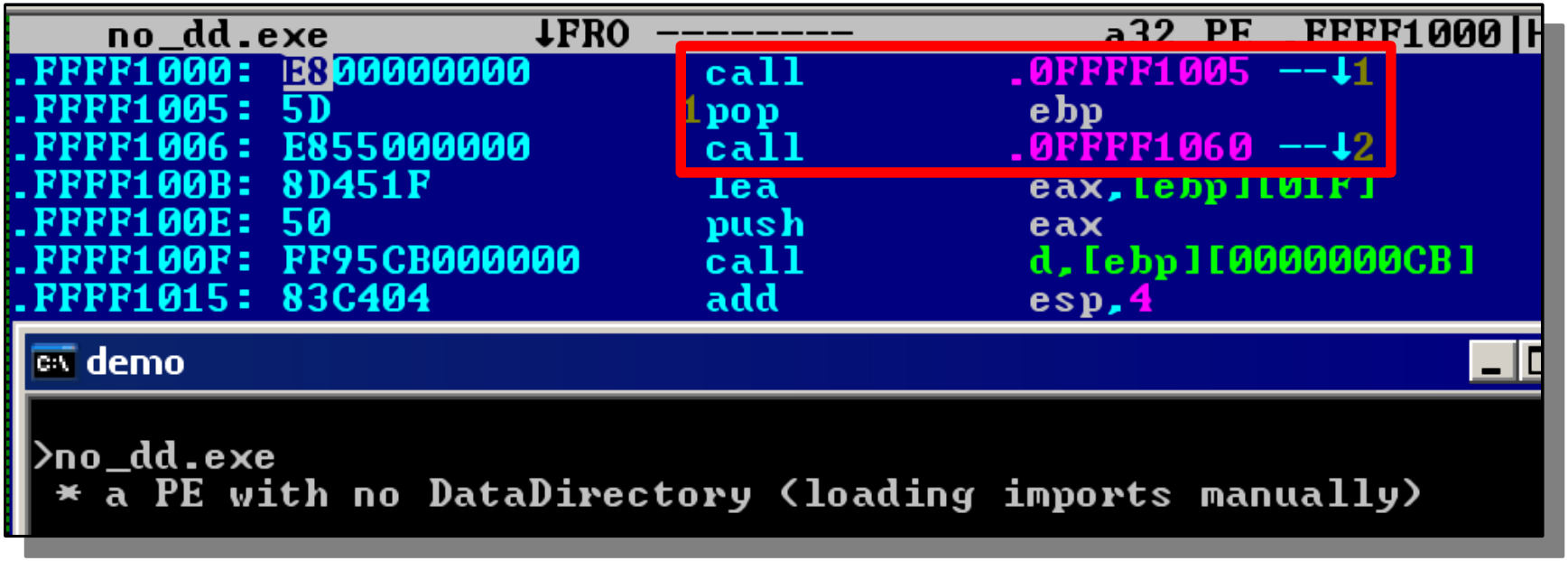
Relocations tricks

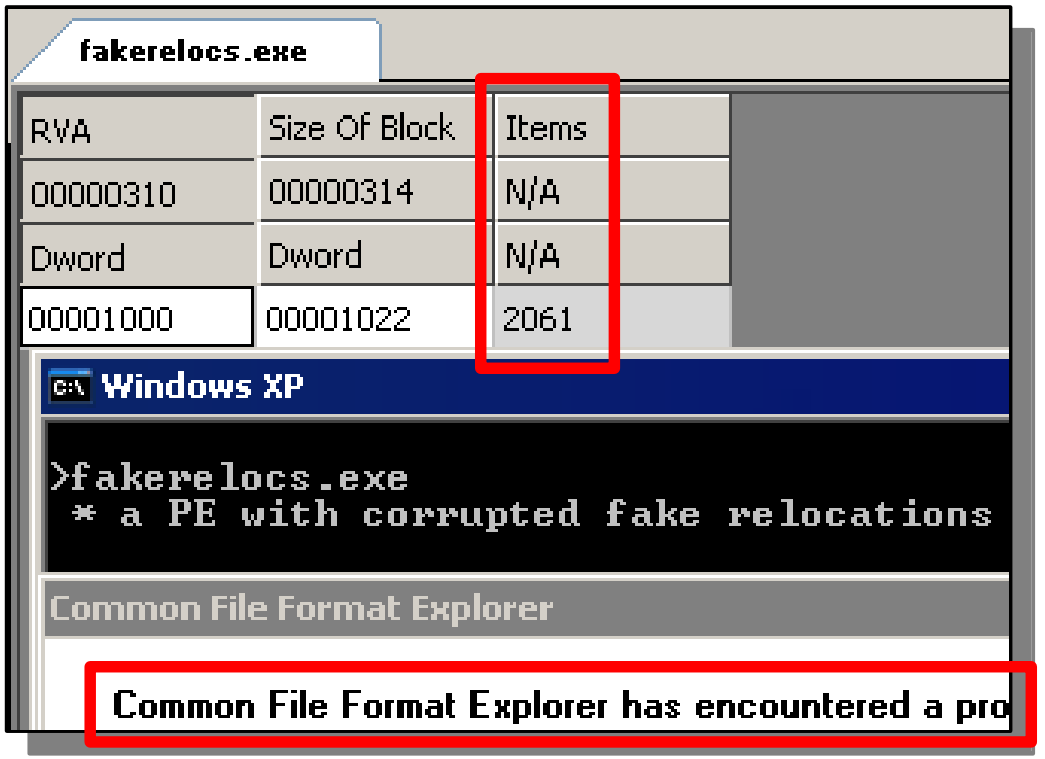
- allows any ImageBase
  - required on VAs: code, TLS, .Net
- ignored if not required
  - no ImageBase change (→ fake relocs!)
  - no code
  - 64 bits RIP-relative code
  - IP-independant code
- can relocate anything
  - relocate ImageBase → alter EntryPoint



ibknoreloc64

no_dd





fakerelocs

```
...
IMAGEBASE equ 0FFFF0000h
DELTA equ 20000h
...
    at IMAGE_OPTIONAL_HEADER32.AddressOfEntryPoint,      dd EntryPoint - IMAGEBASE + DELTA
reloc00:
    at IMAGE_OPTIONAL_HEADER32.ImageBase,                dd IMAGEBASE
...
```

```
>ibreloc.exe
 * relocated ImageBase (only affects EntryPoint)
>
```

ibreloc

Relocation types (in theory)

HIGHLOW

- standard ImageBase delta

ABSOLUTE

- do nothing
- just for alignment padding

Relocation types in practice

- type 6 and 7 are entirely skipped
  - type 8 is forbidden
- type 4 (HIGHADJ) requires an parameter
  - that is actually not taken into account (bug)
- type 2 (LOW) doesn't do anything
  - because ImageBase are 64kb aligned
- type MIPS and IA64 are present on all archs
- at last, some cleanup in Windows 8!

```
switch_table_xp dd offset _next_reloc    ; DATA XREF
                                         ; absolute
                dd offset high           ; high
                dd offset low            ; low
                dd offset highlow        ; highlow
                dd offset highadj        ; highadj
                dd offset mips           ; mips
                dd offset _next_reloc    ; section
                dd offset _next_reloc    ; rel32
                dd offset return_0       ; type 8
                dd offset ia64           ; ia64
                dd offset dir64          ; dir64
```

```
switch_table_w7 dd offset _next_reloc    ; DATA XREF:
                                         ; absolute
                dd offset high           ; high
                dd offset low            ; low
                dd offset highlow        ; highlow
                dd offset highadj        ; highadj
                dd offset mips           ; mips
                dd offset _next_reloc    ; section
                dd offset _next_reloc    ; rel32
                dd offset _return_0      ; type 8
                dd offset ia64           ; mips16/ia64
                dd offset dir64          ; dir64
```

```
[...]
        sub eax, 0
        jz  return
        dec eax
        jz  high
        dec eax
        jz  low
        dec eax
        jnz more_than_highlow

high_low: ; TYPE
        mov eax, [ebp+delta]
        add [lpReloc], eax

return:
        pop edi
        mov eax, esi
        pop esi
        pop ebp
        retn    10h

[...]

low: ; TYPE
[...]
        jmp return

more_than_highlow:
        dec eax
        jz  highadj
        sub eax, 6
        jz  dir64
        xor esi, esi
        jmp return

dir64: ; TYPE
[...]
        jmp return

highadj: ; TYPE
[...]
        jmp return

high: ; TYPE
[...]
        jmp return
```

relocations' archeology

- HIGHADJ was there all along
- MIPS was recognized but rejected by Win95
- NT3.1 introduces MIPS – available in all archs.
- LOW was rejected by Win95/WinME
  - while it does nothing on other versions
- Windows 2000 had an extra relocation type, also with a parameter

Bonus:

Win95 relocations use 2 copies of the exact same code.
code optimization FTW!

```
win95_relocs    dd offset absolute      ; DATA
                dd offset high
                dd offset return_m8     ; low
                dd offset high_low
                dd offset highadj
                dd offset return_m8_0   ; mips
```

```
relocs_nt31     dd offset _next_reloc
                dd offset high
                dd offset low
                dd offset highlow
                dd offset highadj
                dd offset mips
```

```
WinME_relocs    dd offset absolute      ; DATA
                dd offset high
                dd offset return_m7     ; low
                dd offset high_low
                dd offset highadj
```

```
switch_table_2k dd offset _next         ; DATA XREF:
                                        ; absolute
                dd offset high          ; high
                dd offset low           ; low
                dd offset highlow       ; highlow
                dd offset highadj       ; highadj
                dd offset mips          ; mips
                dd offset _next         ; section
                dd offset _next         ; rel32
                dd offset _return_0     ; 8
                dd offset ia64          ; mips16/ia64
                dd offset highlow       ; dir64
                dd offset high3adj      ; high3adj
```

messing with relocations

- 4 relocation types actually do nothing
- All relocations can be applied on a bogus address
  - HighAdj's parameter used as a trick
- Relocations can alter relocations
  - one block can alter the next
- Relocations can decrypt data
  - set a kernel ImageBase
  - default ImageBase is known
- No static analysis possible
  - but highly suspicious :D

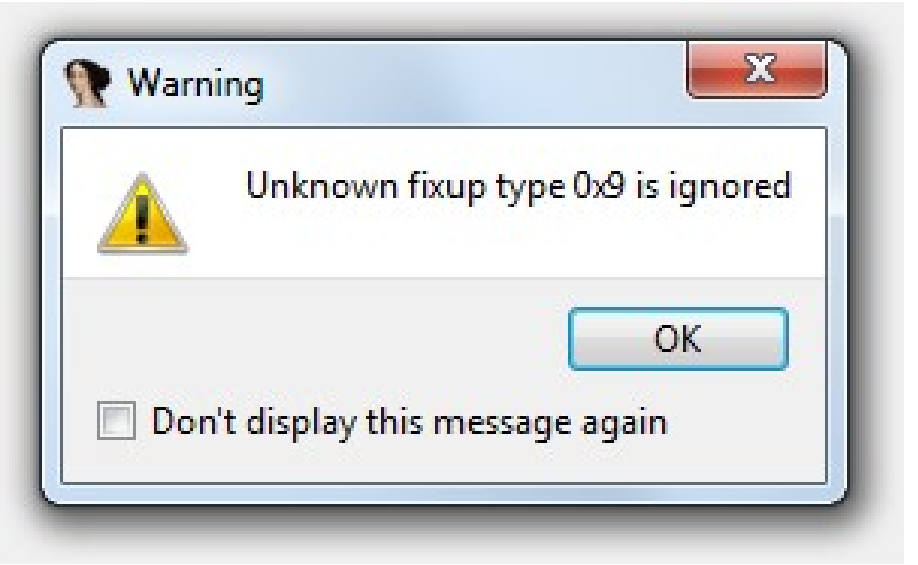
Item	RVA	Type
Word	N/A	N/A
0000	00001002	ABSOLUTE
2000	00001002	LOW
6000	00001002	UNKNOWN
7000	00001002	UNKNOWN

Item	RVA	Type
Word	N/A	N/A
0000	000010B7	ABSOLUTE
1000	000010B7	HIGH
2000	000010B7	LOW
3000	000010B7	HIGHLOW
4000	000010B7	HIGHADJ
8000	000010B7	UNKNOWN
5000	000010B7	MIPS_JMPA...
6000	000010B7	UNKNOWN
7000	000010B7	UNKNOWN
9000	000010B7	IA64_IMM64
A000	000010B7	DIR64

Item	RVA	Type
Word	N/A	N/A
3000	00001142	HIGHLOW
3000	00001142	HIGHLOW
1000	00001142	HIGH
5000	00001142	MIPS_JMPA...

RVA	Size Of Block	Items
00000300	00000304	N/A
Dword	Dword	N/A
00001002	00000010	4
000010B7	0000001E	11
00001142	00000010	4
00001002	03FB800C	33406978

```
seg000:00011002
seg000:00011002                 public start
seg000:00011002 start:
seg000:00011002                 inc     eax
seg000:00011003                 and     [edx+0], dl
seg000:00011005                 add     [eax], eax
seg000:00011007                 add     [ecx], eax
seg000:00011009                 retf    112h
seg000:00011009 ; ---------------------------------
seg000:0001100C                 db 0, 1, 0
seg000:0001100F                 dw 202h
seg000:00011011                 dw 302h
seg000:00011013                 db 1
seg000:00011014                 dw 12C2h
seg000:00011016                 dw 1
```




```
00011002 public start
00011002 start:
00011002 push    offset aDecryptionViaR
00011007 call    printf
0001100D add     esp, 4
00011010 push    0
00011012 call    ExitProcess
00011012 ; ---------------------------
```

Code in the header

- header is executable
  - packers put some data or jumps there
- many unused fields
- many less important fields
  - Peter Ferrie
    http://pferrie.host22.com/misc/pehdr.htm

→ real code in the header

PE Header

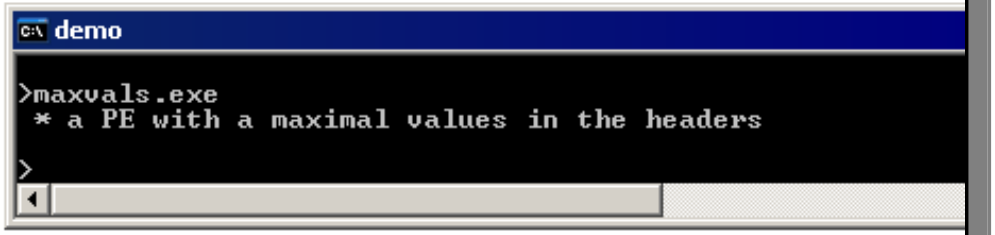
Signature	PE
Machine	0x14c
NumberOfSections	1
TimeDateStamp	0xffffffff
PointerToSymbolTable	0xffffffff
NumberOfSymbols	0xffffffff
SizeOfOptionalHeader	0xe0
Characteristics	0xdfff
Magic	0x10b
LinkerVersion	255.255
SizeOfCode	0xffffffff
SizeOfInitializedData	0xffffffff
SizeOfUninitializedData	0xffffffff
AddressOfEntryPoint	0x1000
BaseOfCode	0xffffffff
BaseOfData	0xffffffff
ImageBase	0x400000
SectionAlignment	0x1000
FileAlignment	0x200
OperatingSystemVersion	65535.65535
ImageVersion	65535.65535
SubsystemVersion	4.65535
Reserved1	0xffffffff
SizeOfImage	0x2000
SizeOfHeaders	0x160
CheckSum	0xffffffff
Subsystem	3
DllCharacteristics	0xfa7f
SizeOfStackReserve	0xffffff
SizeOfStackCommit	0x1fff
SizeOfHeapReserve	0xffffff
SizeOfHeapCommit	0x1fff
LoaderFlags	0xffffffff
NumberOfRvaAndSizes	0xffffffff

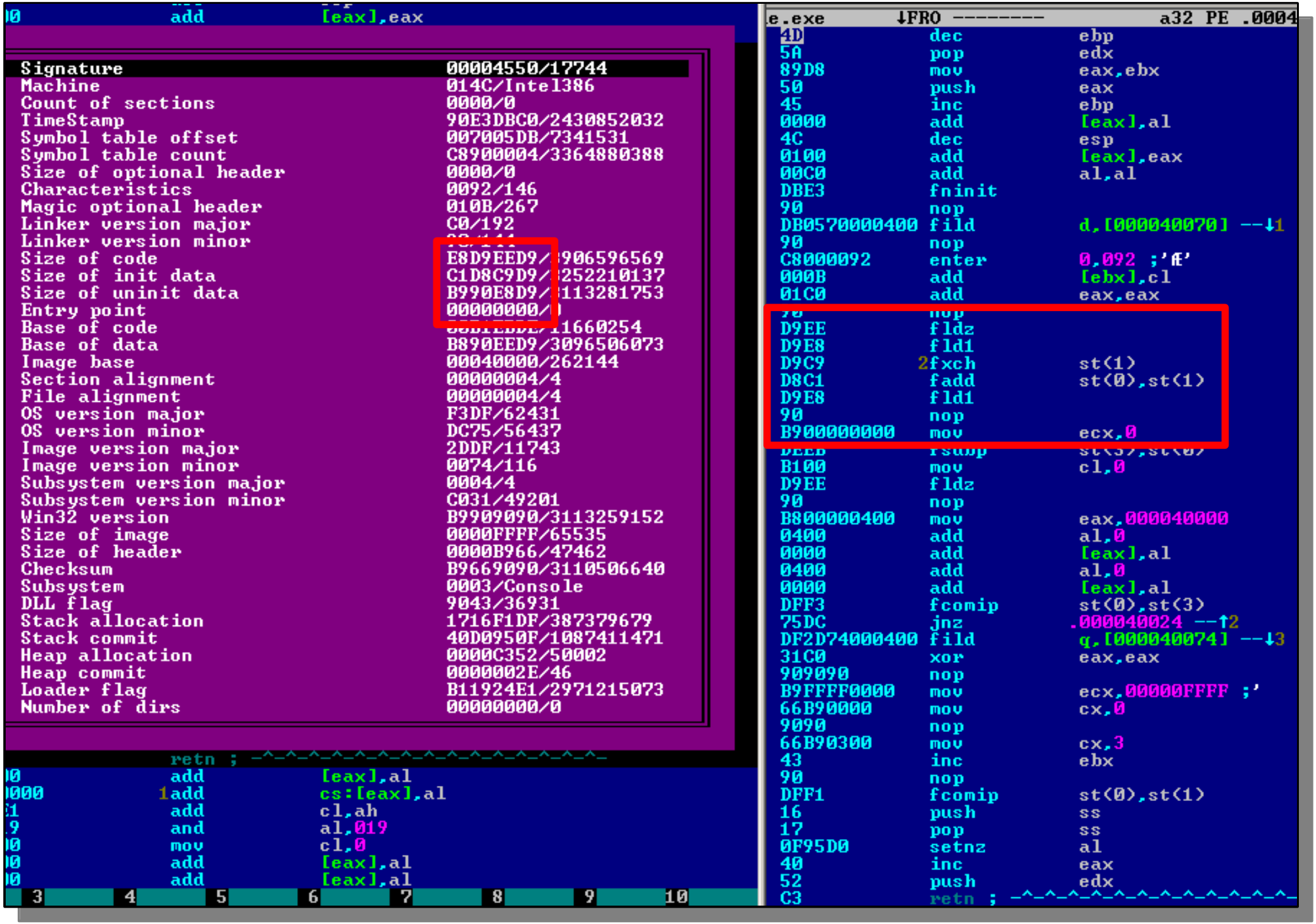
Sections

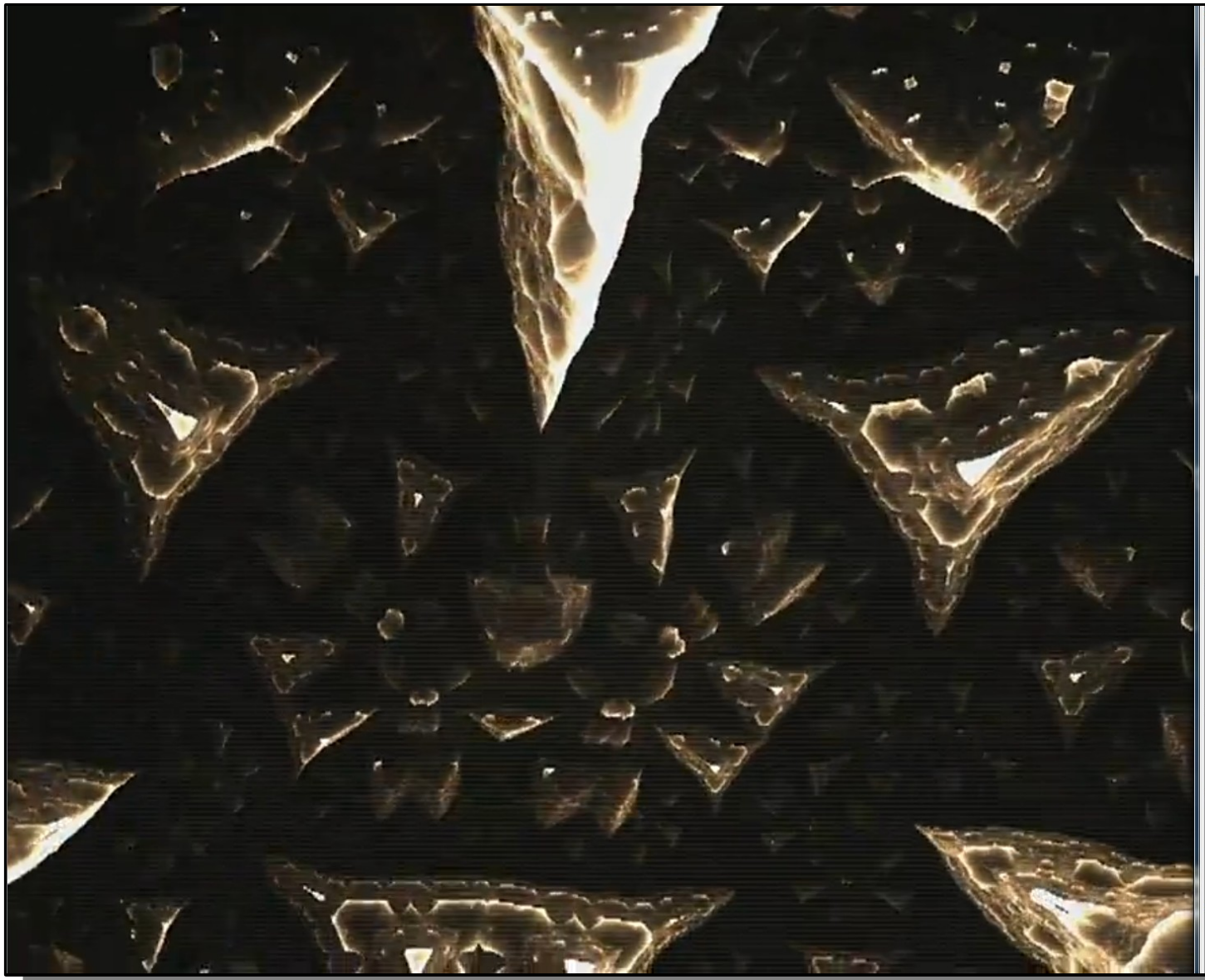
name	va	vsize	raw size	flags
������	0x1000	0x1000	0x200	RWX CODE IDATA UDATA DISCARDABLE SHARED

Data Directory

type	va	size
EXPORT	0xffffffff	0xffffffff
IMPORT	0x1050	0xffffffff
RESOURCE	0	0xffffffff
EXCEPTION	0xffffffff	0xffffffff
SECURITY	0xffffffff	0xffffffff
BASERELOC	0xffffffff	0xffffffff
DEBUG	0xffffffff	0
ARCHITECTURE	0xffffffff	0xffffffff
GLOBALPTR	0xffffffff	0xffffffff
TLS	0	0xffffffff
LOAD_CONFIG	0	0xffffffff
Bound_IAT	0	0xffffffff
IAT	0	0xffffffff
Delay_IAT	0xffffffff	0xffffffff
CLR_Header	0	0xffffffff
	0xffffffff	0xffffffff




add [eax],eax
Signature 00004550/17744
Machine 014C/Intel386
Count of sections 0000/0
TimeStamp 90E3DBC0/2430852032
Symbol table offset 007005DB/7341531
Symbol table count C8900004/3364880388
Size of optional header 0000/0
Characteristics 0092/146
Magic optional header 010B/267
Linker version major C0/192
Linker version minor
Size of code E8D9EED9/3906596569
Size of init data C1D8C9D9/3252210137
Size of uninit data B990E8D9/3113281753
Entry point 00000000/0
Base of code 11660254
Base of data B890EED9/3096506073
Image base 00040000/262144
Section alignment 00000004/4
File alignment 00000004/4
OS version major F3DF/62431
OS version minor DC75/56437
Image version major 2DDF/11743
Image version minor 0074/116
Subsystem version major 0004/4
Subsystem version minor C031/49201
Win32 version B9909090/3113259152
Size of image 0000FFFF/65535
Size of header 0000B966/47462
Checksum B9669090/3110506640
Subsystem 0003/Console
DLL flag 9043/36931
Stack allocation 1716F1DF/387379679
Stack commit 40D0950F/1087411471
Heap allocation 0000C352/50002
Heap commit 0000002E/46
Loader flag B11924E1/2971215073
Number of dirs 00000000/0
retn ; _^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_
add [eax],al
add cs:[eax],al
add cl,ah
and al,019
mov cl,0
add [eax],al
add [eax],al
3 4 5 6 7 8 9 10
e.exe ↓FRO -------- a32 PE .0004
dec ebp
pop edx
mov eax,ebx
push eax
inc ebp
add [eax],al
dec esp
add [eax],eax
add al,al
fninit
nop
fild d,[0000400070] --↓1
nop
enter 0,092 ;'Æ'
add [ebx],cl
add eax,eax
nop
fldz
fld1
fxch st(1)
fadd st(0),st(1)
fld1
nop
mov ecx,0
mov cl,0
fldz
nop
mov eax,000040000
add al,0
add [eax],al
add al,0
add [eax],al
fcomip st(0),st(3)
jnz .000040024 --↑2
fild q,[0000400074] --↓3
xor eax,eax
nop
mov ecx,00000FFFF ;'
mov cx,0
nop
mov cx,3
inc ebx
nop
fcomip st(0),st(1)
push ss
pop ss
setnz al
inc eax
push edx
retn ; _^_^_^_^_^_^_^_^_^_^_^_^_^_^_^_



```
db 'MZ' ;fixed

aRa db 'RA' ; ugly pun :p

aPe db 'PE',0,0 ; Signature ; fixed
    dw 14Ch         ; Machine ; fixed
    dw 0            ; NumberOfSections ; controlled, better if 0

loc_01:
    bt  [dat], ebp  ; timestamp ; fully controlled
    adc esi, esi
    jmp loc_04

    dw 8            ; SizeOfOptionalHeader ; almost fixed
    dw 2            ; characteristics ; almost fixed ; FILE_IS_EXECUTABLE
    dw 108h         ; Magic ; fixed

loc_02:
    xor ecx, ecx    ; majorlinker ; fully controlled
    shr bl, 1       ; Code  ; fully controlled
    jb  l_91
    jnz l_99 ; initData  ; fully controlled

loc_03:
    popa
    jnz l_A1 ; unitialized data is in the middle ;  fully controlled
    inc eax
    inc edx
    mov ebp, EntryPoint - IMAGEBASE ; bogus operation on EBP to encode EntryPoint ; very strict
    ror byte [esi], cl ; BaseCode  ; fully controlled
    jb  l_9D
    shr edx, 1   ; BaseData  ; fully controlled
    jmp l_9F

    dd IMAGEBASE ; IMAGEBASE ; very strict
    dd 4   ; SectionAlig ; very strict
    dd 4   ; FileAlig ; very strict ; also serves as DOS_HEADER.e_lfanew

loc_04:
    add eax, eax ; MajorOS ; fully controlled

l_46:
    test eax, eax ; MinorOS ; fully controlled
    jns loc_01 ; MajorImage ; fully controlled
    jmp l_54 ; MinorImage ; fully controlled ; intermediate jump to enable a short jump to l_CF

    dw 3   ; MajorSubsystemVersion  ; very strict
    dw 8000h  ; MinorSubsystemVersion ; strict
    dd 0   ; Win32VersionValue ; could be anything but would break D3D support :(

l_54:
    jmp l_CF ; SizeOfImage ; jmp encoded on the lower word
        dw 100h ; upper word of SizeOfImage

    dd 30h   ; SizeOfHeaders ; strict

l_5C:
    jle l_46 ; Checksum ; fully controlled
    pop eax
    retn

    dw 2   ; subsystem ; fixed
    dw 0   ; characteristics ; almost controlled

EntryPoint:
    nop    ; StackReserve ; almost controlled
    mov edi, TARGET ; stack commit ; almost controlled
    push 1
    pop eax  ; HeapReserve ; almost controlled
```

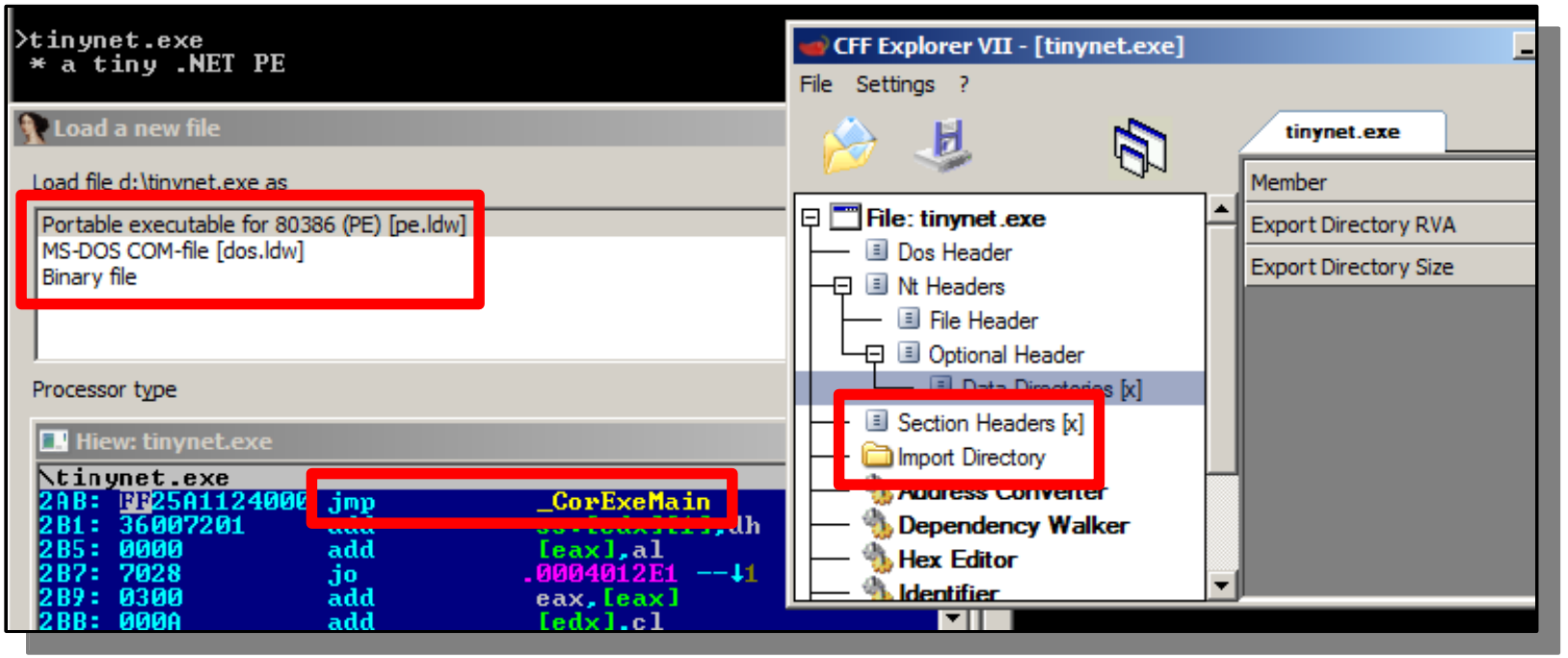
.Net

Loading process:

1. PE loader
   - requires only imports (DD[1]) at this stage
2. MSCoree.dll called
3. .Net Loader
   - requires CLR (DD[13]) and relocations (DD[5])
   - forgets to check NumberOfRvaAndSizes :(
     - works with NumberOfRvaAndSizes = 2

fails IDA, reflector – but already in the wild

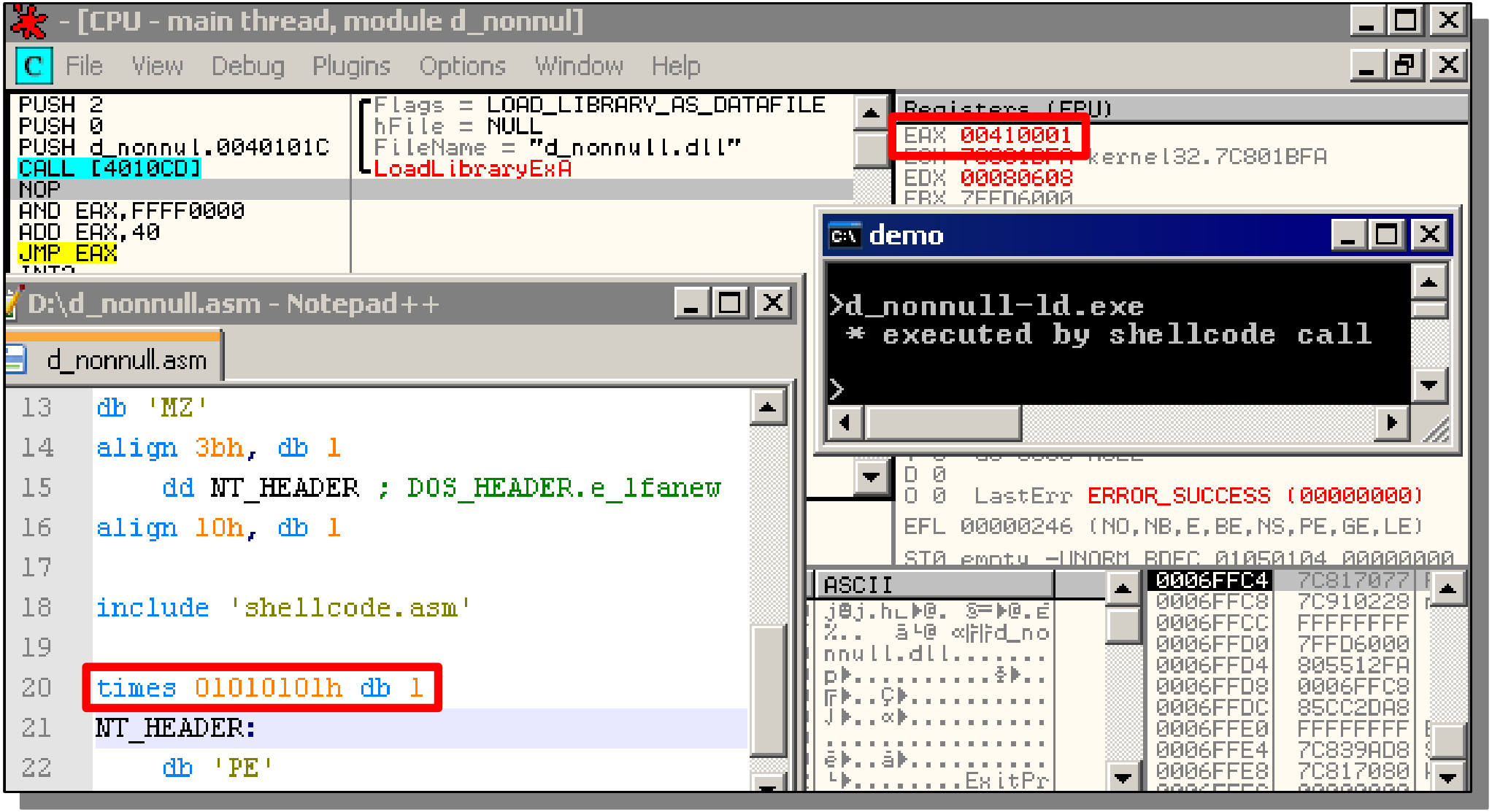
PE	.NET
…	…
imports	…
…	…
…	…
…	…
…	relocs
…	…
…	…
…	…
…	…
…	…
…	…
…	…
…	…
…	CLR
…	…



non-null PE

- LoadlibraryEx with LOAD_LIBRARY_AS_DATAFILE
- data file PE only needs MZ, e_lfanew, 'PE\0\0'
- 'PE' at the end of the file
  - pad enough so that e_lfanew doesn't contain 00s

a non-null PE can be created and loaded


- [CPU - main thread, module d_nonnul]
File View Debug Plugins Options Window Help
PUSH 2
PUSH 0
PUSH d_nonnul.0040101C
CALL [4010CD]
NOP
AND EAX,FFFF0000
ADD EAX,40
JMP EAX
Flags = LOAD_LIBRARY_AS_DATAFILE
hFile = NULL
FileName = "d_nonnull.dll"
LoadLibraryExA
Registers (FPU)
EAX 00410001
EDX 00080608
kernel32.7C801BFA
demo
>d_nonnull-ld.exe
* executed by shellcode call
D:\d_nonnull.asm - Notepad++
d_nonnull.asm
13 db 'MZ'
14 align 3bh, db 1
15 dd NT_HEADER ; DOS_HEADER.e_lfanew
16 align 10h, db 1
17
18 include 'shellcode.asm'
19
20 times 01010101h db 1
21 NT_HEADER:
22 db 'PE'
LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ASCII
nnull.dll.....
ExitPr
0006FFC4 7C817077
0006FFC8 7C910228
0006FFCC FFFFFFFF
0006FFD0 7FFD6000
0006FFD4 805512FA
0006FFD8 0006FFC8
0006FFDC 85CC2DA8
0006FFE0 FFFFFFFF
0006FFE4 7C839AD8
0006FFE8 7C817080

Resources-only DLL

- 1 valid section
  - 65535 sections under XP!
- 1 DataDirectory

PE Header

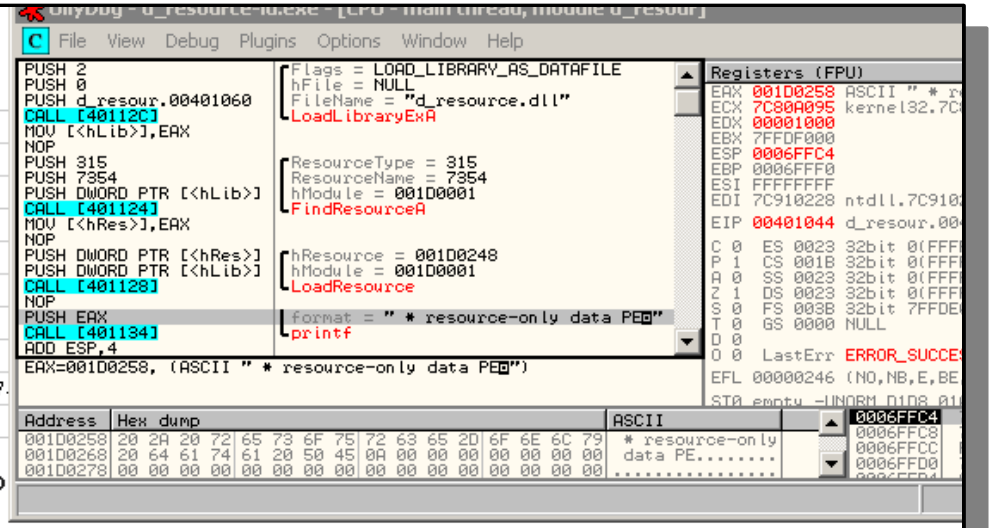
Signature	PE
Machine	0x1111
NumberOfSections	0xffff
TimeDateStamp	0xffffffff
PointerToSymbolTable	0xffffffff
NumberOfSymbols	0xffffffff
SizeOfOptionalHeader	0xe0
Characteristics	0x1111
Magic	0x10b
LinkerVersion	255.255
SizeOfCode	0xffffffff
SizeOfInitializedData	0xffffffff
SizeOfUninitializedData	0xffffffff
AddressOfEntryPoint	0xffffffff
BaseOfCode	0xffffffff
BaseOfData	0xffffffff
ImageBase	0xffffffff
SectionAlignment	0xffffffff
FileAlignment	0xffffffff
OperatingSystemVersion	65535.65535
ImageVersion	65535.65535
SubsystemVersion	65535.65535
Reserved1	0xffffffff
SizeOfImage	0xffffffff
SizeOfHeaders	0x1000
CheckSum	0xffffffff
Subsystem	0xffff
DllCharacteristics	0xffff
SizeOfStackReserve	0xffffffff
SizeOfStackCommit	0xffffffff
SizeOfHeapReserve	0xffffffff
SizeOfHeapCommit	0xffffffff
LoaderFlags	0xffffffff

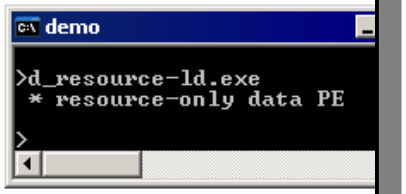
Sections

name	
Ts▯▯O▯▯▯	
	0x7.

Data Directo

type	va	size
EXPORT	0xffffffff	0xffffffff
IMPORT	0xffffffff	0xffffffff
RESOURCE	0x1000	0xffffffff
EXCEPTION	0xffffffff	0xffffffff
SECURITY	0xffffffff	0xffffffff
BASERELOC	0xffffffff	0xffffffff
DEBUG	0xffffffff	0xffffffff
ARCHITECTURE	0xffffffff	0xffffffff
GLOBALPTR	0xffffffff	0xffffffff
TLS	0xffffffff	0xffffffff
LOAD_CONFIG	0xffffffff	0xffffffff
Bound_IAT	0xffffffff	0xffffffff
IAT	0xffffffff	0xffffffff
Delay_IAT	0xffffffff	0xffffffff
CLR_Header	0xffffffff	0xffffffff
	0xffffffff	0xffffffff

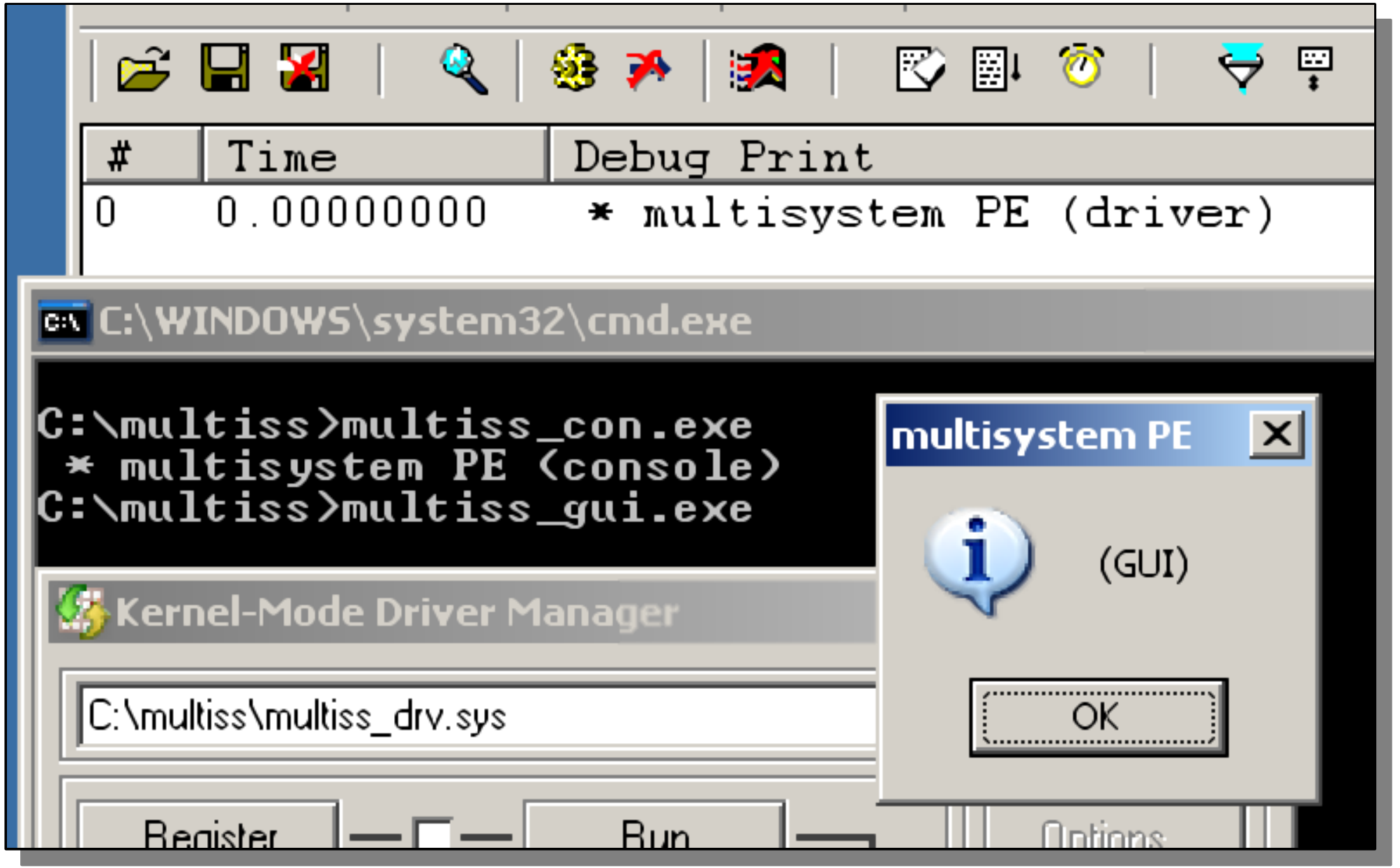




subsystems

- no fundamental differences
  - low alignments for drivers
  - incompatible imports: NTOSKRNL ↔ KERNEL32
  - console ↔ gui : IsConsoleAttached

→ a PE with low alignments and no imports can work in all 3 subsystems

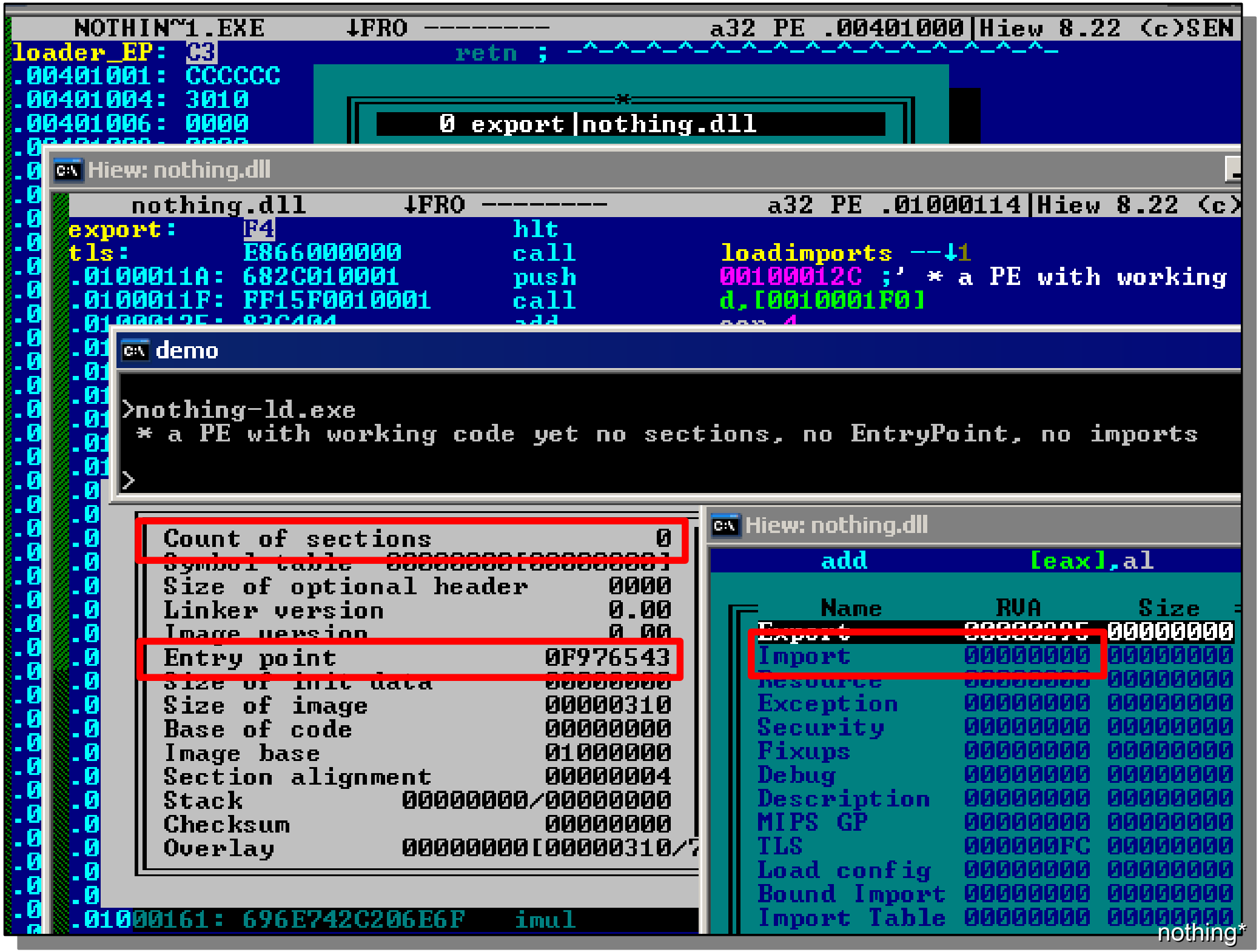

#
Time
Debug Print
0
0.00000000
* multisystem PE (driver)
C:\WINDOWS\system32\cmd.exe
C:\multiss>multiss_con.exe
* multisystem PE (console)
C:\multiss>multiss_gui.exe
multisystem PE
(GUI)
OK
Kernel-Mode Driver Manager
C:\multiss\multiss_drv.sys
Register
Run
Options

a 'naked' PE with code

- low alignments → no section
- no imports → resolve manually APIs
- TLS only → no EntryPoint

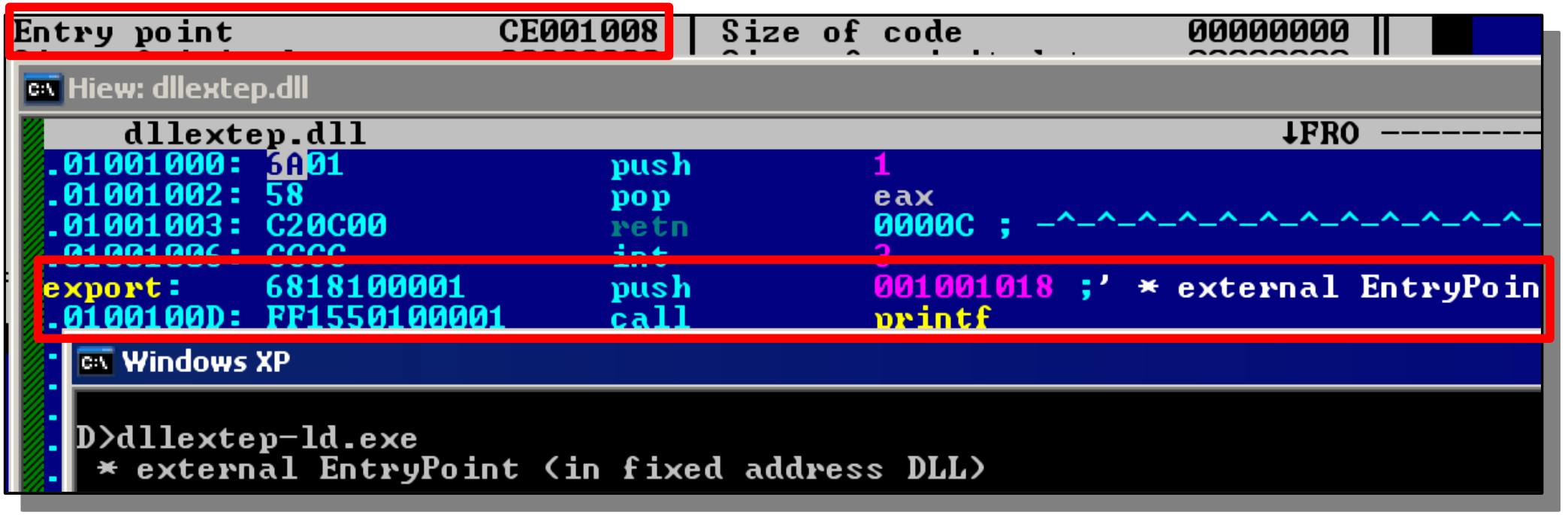
no EntryPoint, no section, no imports,

but executed code


NOTHIN~1.EXE ↓FRO -------- a32 PE .00401000|Hiew 8.22 (c)SEN
loader_EP: C3 retn ; -^-
.00401001: CCCCCC
.00401004: 3010
.00401006: 0000
0 export|nothing.dll
Hiew: nothing.dll
nothing.dll ↓FRO -------- a32 PE .01000114|Hiew 8.22 (c)
export: F4 hlt
tls: E866000000 call loadimports --↓1
.0100011A: 682C010001 push 0100012C ;' * a PE with working
.0100011F: FF15F0010001 call d,[010001F0]
demo
>nothing-ld.exe
* a PE with working code yet no sections, no EntryPoint, no imports
>
Count of sections 0
Size of optional header 0000
Linker version 0.00
Entry point 0F976543
Size of image 00000310
Base of code 00000000
Image base 01000000
Section alignment 00000004
Stack 00000000/00000000
Checksum 00000000
Overlay 00000000[00000310/7
.01000161: 696E742C206E6F imul
Hiew: nothing.dll
add [eax],al
Name RVA Size
Import 00000000 00000000
Exception 00000000 00000000
Security 00000000 00000000
Fixups 00000000 00000000
Debug 00000000 00000000
Description 00000000 00000000
MIPS GP 00000000 00000000
TLS 000000FC 00000000
Load config 00000000 00000000
Bound Import 00000000 00000000
Import Table 00000000 00000000
nothing*

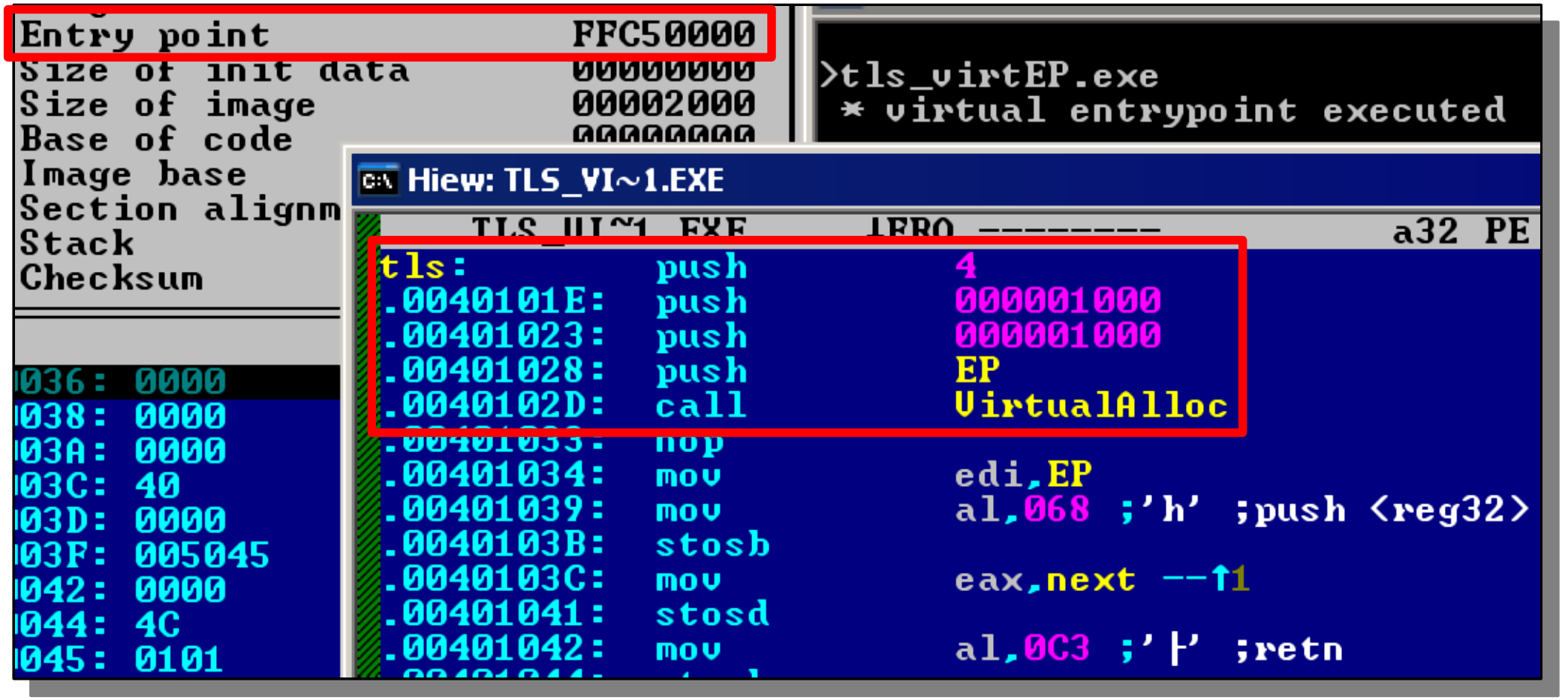
external EntryPoint (1/2)

- in a DLL (with no relocations)



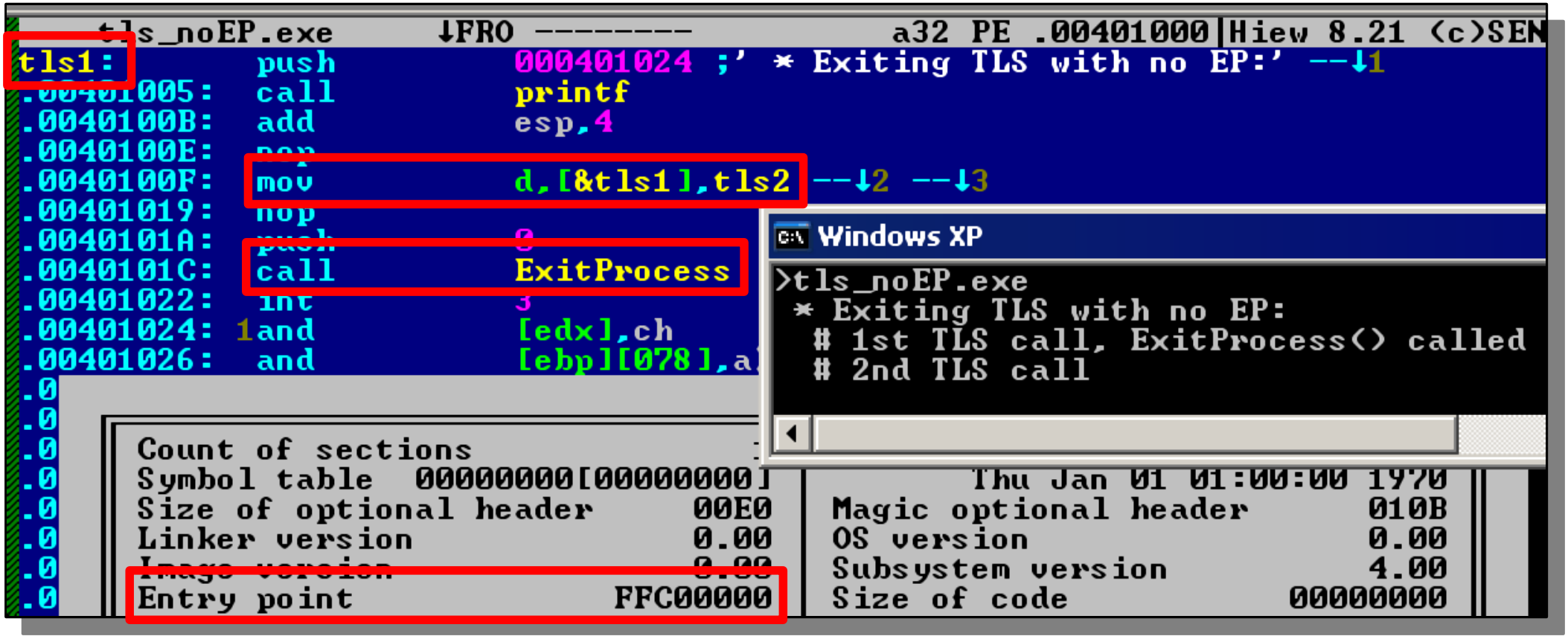
external EntryPoint (2/2)

- allocated just before in a TLS



skipped EntryPoint

ignored via terminating TLS

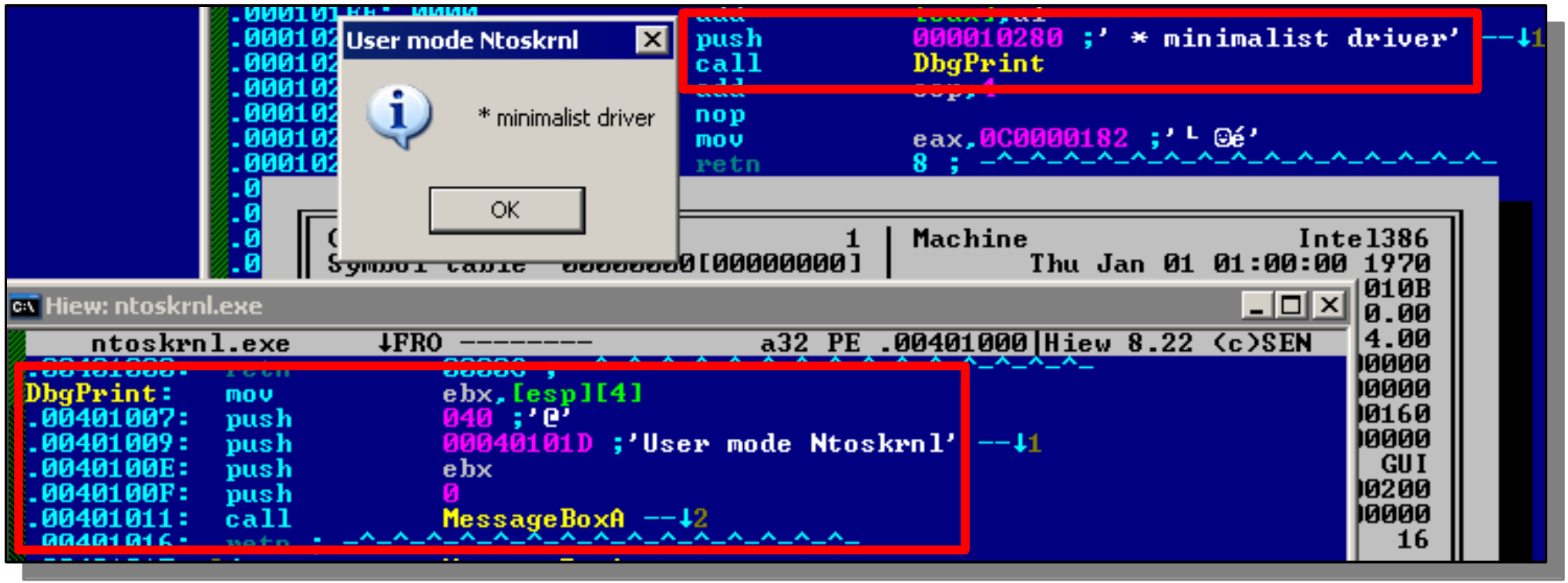


from ring 0 to ring 3

- kernel debugging is heavy
  - kernel packers are limited

1. change subsystem
2. use fake kernel DLLs (ntoskrnl, etc...)
   - redirect APIs
     - DbgPrint → MessageBoxA, ExAllocatePool → VirtualAlloc

→ automate kernel unpacking

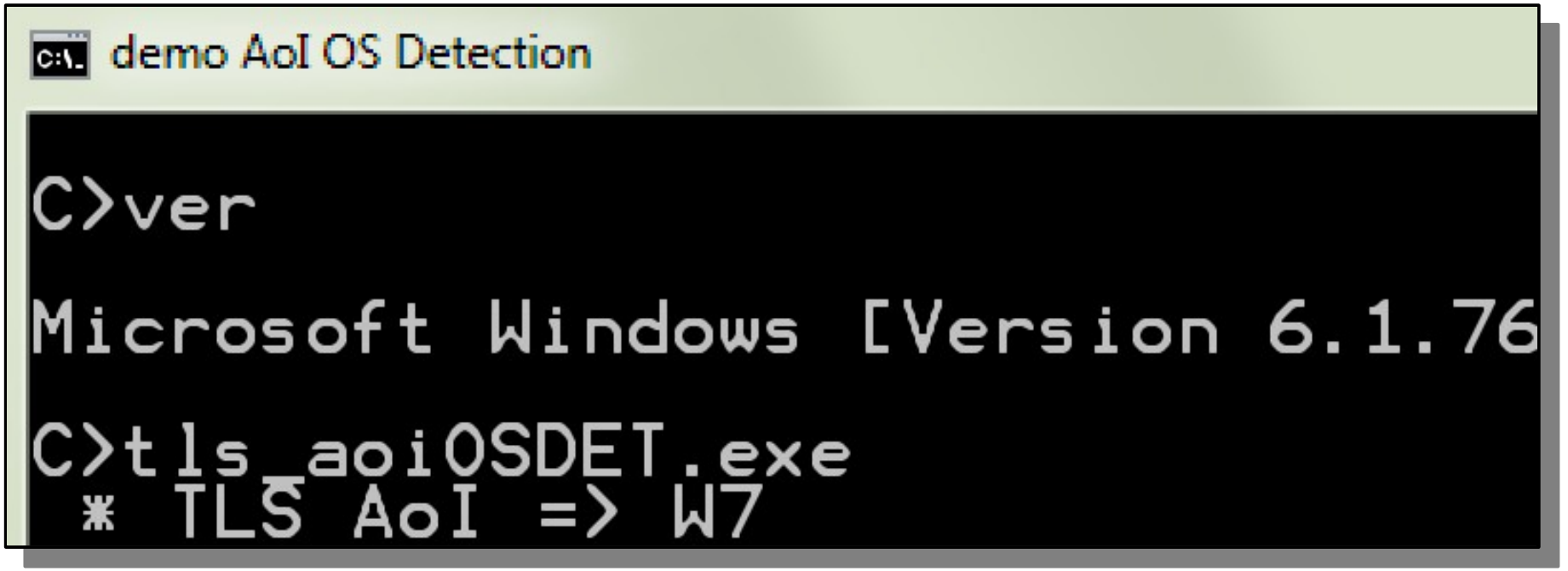

User mode Ntoskrnl
* minimalist driver
OK
push 000010280 ;' * minimalist driver'
call DbgPrint
nop
mov eax,0C0000182 ;'└ Θé'
retn 8 ;
Machine Intel386
Thu Jan 01 01:00:00 1970
Hiew: ntoskrnl.exe
ntoskrnl.exe ↓FRO -------- a32 PE .00401000 Hiew 8.22 (c)SEN
DbgPrint: mov ebx,[esp][4]
.00401007: push 040 ;'@'
.00401009: push 00040101D ;'User mode Ntoskrnl'
.0040100E: push ebx
.0040100F: push 0
.00401011: call MessageBoxA
GUI
16

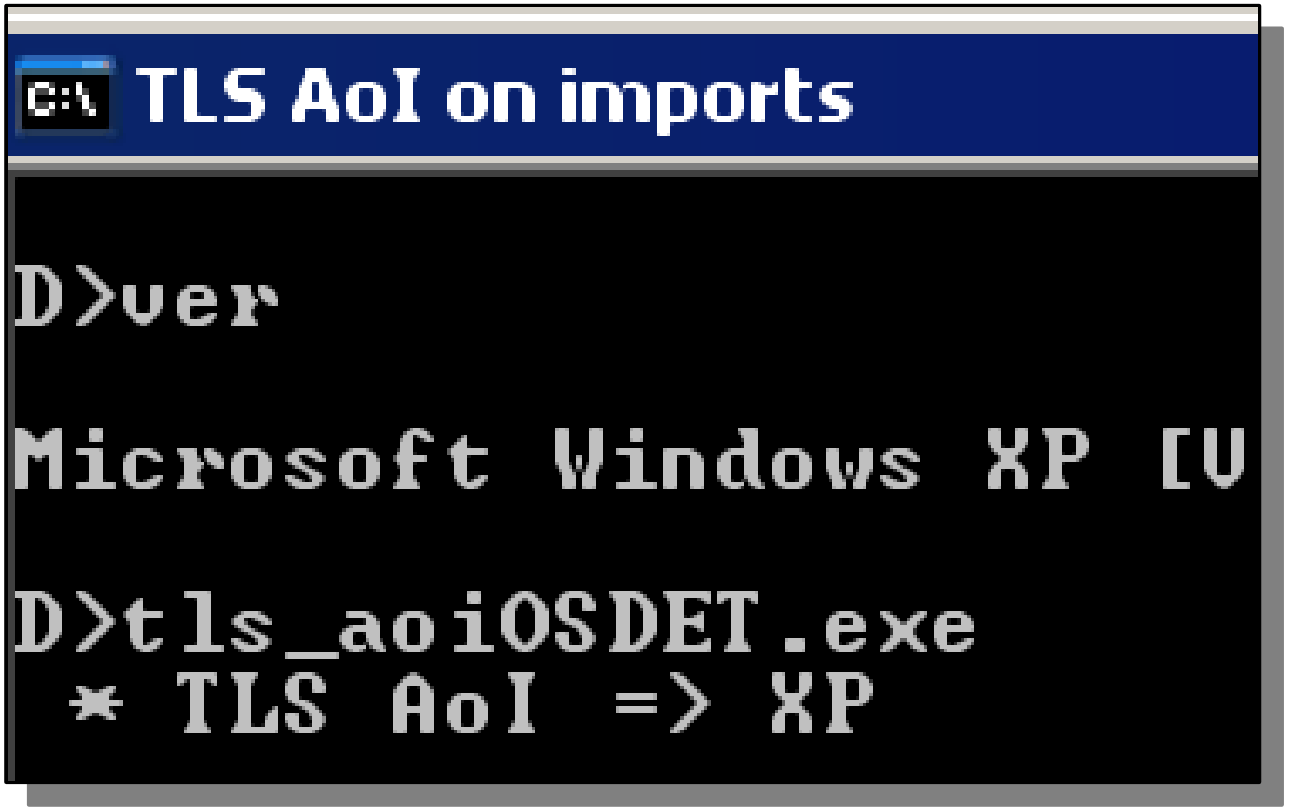
TLS AddressOfIndex

- pointer to dword
- overwritten with 0, 1... on n^{th} TLS loading
- easy dynamic trick

  call <garbage> on file → *call $+5* in memory
- handled before imports under XP, not in W7

same working PE, different loading process





tls_aoiOSDET

Manifest

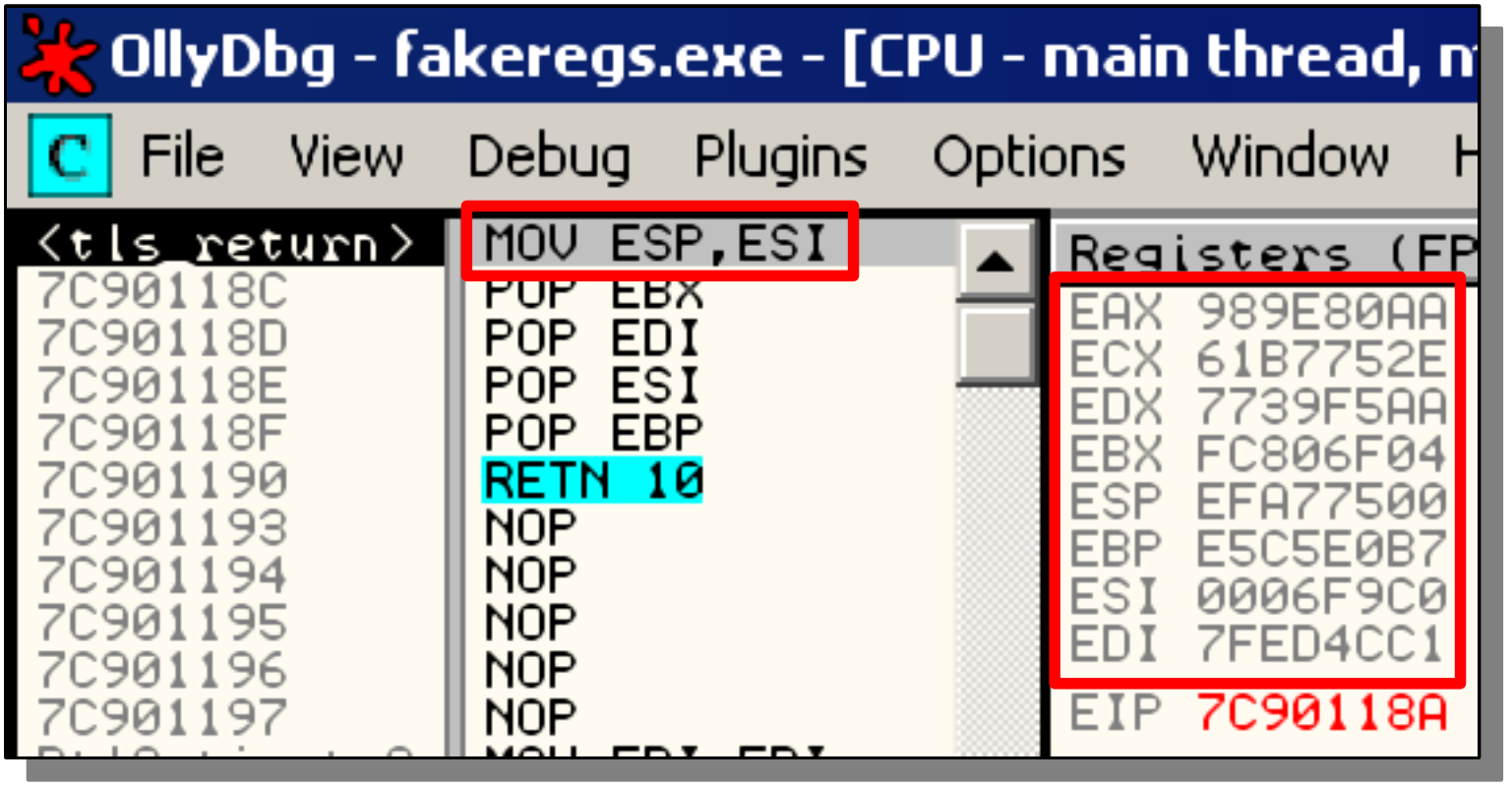
- XML resource
  - can fail loading
  - can crash the OS ! (KB921337)
- Tricky to classify
  - ignored if wrong type

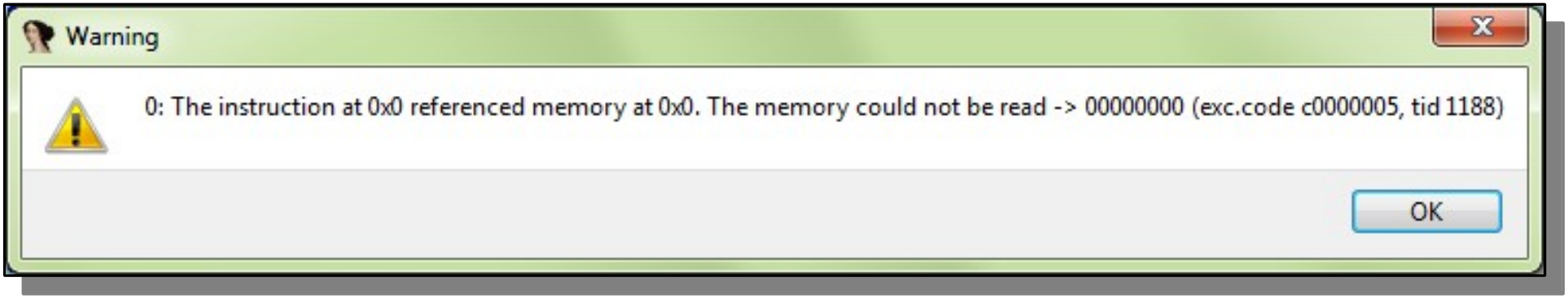
Minimum Manifest

```
<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'/>
```

DllMain/TLS corruption

- DllMain and TLS only requires ESI to be correct
  - Even ESP can be bogus
  - easy anti-emulator
- TLS can terminate with exception
  - no error reported
  - EntryPoint executed normally


OllyDbg - fakeregs.exe - [CPU - main thread, n
File View Debug Plugins Options Window H
<tls return>
7C90118C
7C90118D
7C90118E
7C90118F
7C901190
7C901193
7C901194
7C901195
7C901196
7C901197
MOV ESP,ESI
POP EBX
POP EDI
POP ESI
POP EBP
RETN 10
NOP
NOP
NOP
NOP
NOP
Registers (FP
EAX 989E80AA
ECX 61B7752E
EDX 7739F5AA
EBX FC806F04
ESP EFA77500
EBP E5C5E0B7
ESI 0006F9C0
EDI 7FED4CC1
EIP 7C90118A

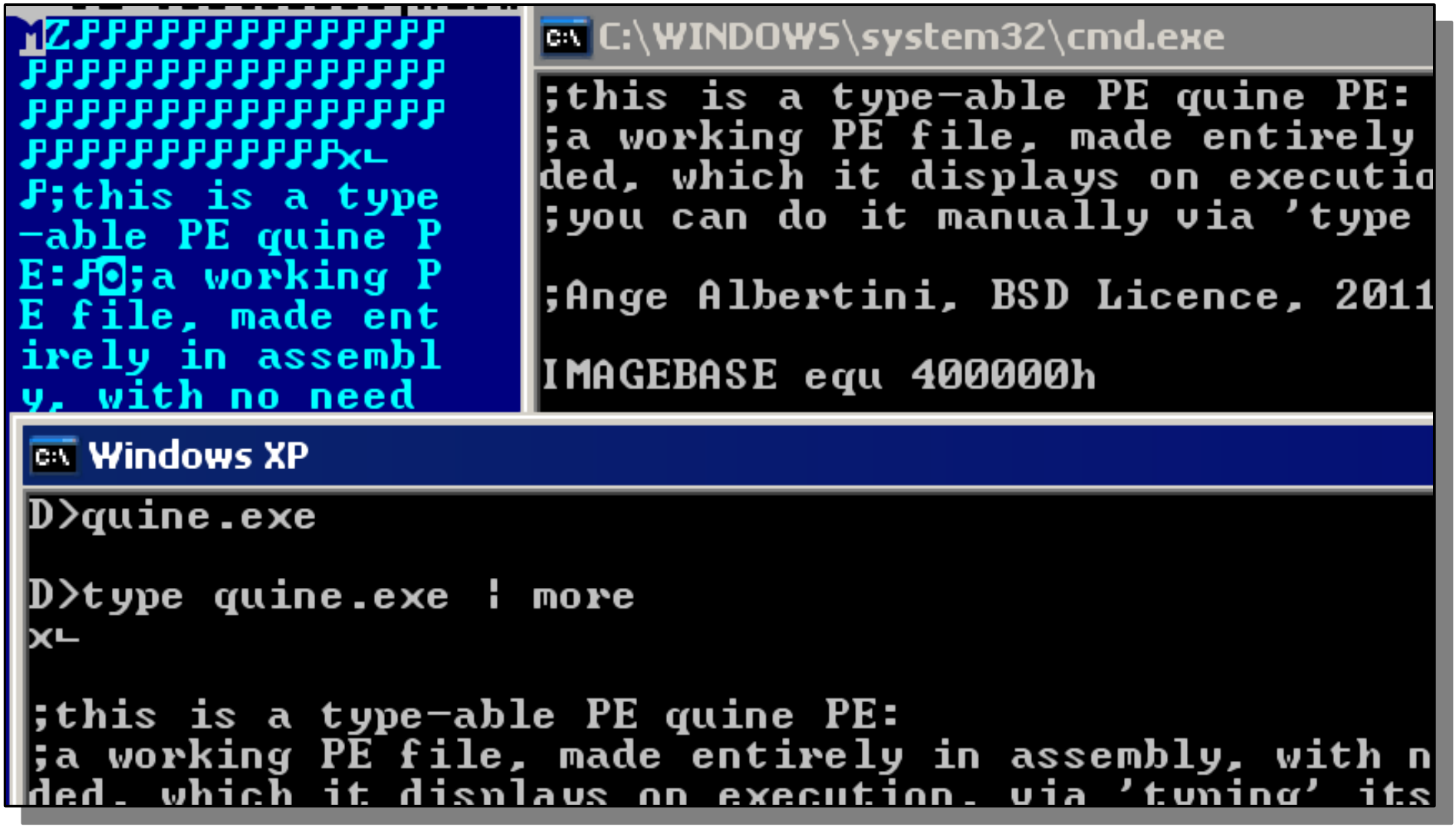

Warning
0: The instruction at 0x0 referenced memory at 0x0. The memory could not be read -> 00000000 (exc.code c0000005, tid 1188)
OK

a Quine PE

- prints its source
  - totally useless – absolutely fun :D

- fills DOS header with ASCII chars
- ASM source between DOS and PE headers

- type-able manually
- types itself in new window when executed


MZ
;this is a type
-able PE quine P
E:
;a working P
E file, made ent
irely in assembl
y, with no need
C:\WINDOWS\system32\cmd.exe
;this is a type-able PE quine PE:
;a working PE file, made entirely
ded, which it displays on executio
;you can do it manually via 'type
;Ange Albertini, BSD Licence, 2011
IMAGEBASE equ 400000h
Windows XP
D>quine.exe
D>type quine.exe | more
;this is a type-able PE quine PE:
;a working PE file, made entirely in assembly, with n
ded, which it displays on execution, via 'typing' its

a binary polyglot

- add %PDF within 400h bytes

  → your PE is also a PDF (→ Acrobat)
- add PK\03\04 anywhere

  → your PE is also a ZIP (→ PKZip)
- throw a Java .CLASS in the ZIP

  → your PE is also a JAR (→ Java)
- add <HTML> somewhere

  → your PE is also an HTML page (→ Mosaic)
- Bonus: Python, JavaScript

Downloads: 1248

Level-Advanced

Description: corkamix.exe is simultaneously a valid:
* Windows Portable Executable binary

corkamix.exe - WinRAR

File Commands Tools Favorites Options Help

Add Extract To Test View Delete Find

corkamix.exe - SFX ZIP archive, unpacked size 542 bytes

Name	S...	P...	Type	CRC32
..			File Folder	
META-INF			File Folder	
corkamix.class	507	507	CLASS File	00000000

Total 1 folder and 507 bytes in

demo

```
>corkamix.exe
CorkaMIX [PE]
>java -jar corkamix.exe
CorkaMIX [Java CLASS in JAR]

>"c:\Program Files\Adobe\Reader 10.0\Reader\AcroRd32.exe" corkamix.exe

>type add_one_byte.py
with open("corkamix.exe", "rb") as source:
        with open("corkamix_.exe", "wb") as target:
                target.write(source.read() + " ")
>add_one_byte.py

>c:\Python27\python corkamix_.exe
CorkaMIX [python]

>copy corkamix.exe corkamix.html
        1 file(s) copied.
```

corkamix.exe - Adobe ...

File Edit View Window Help

1 / 1 15.9%

CorkaMIX [PDF]

Hiew: corkamix.exe

corkamix.exe ↓FRO ---- PE .0 iew 8.22 (c)SEN

MZ=1;print("CorkaMIX [python]"→%PDF-1. obj<<>>strea <html>@ PE L@
@@@@CorkaMIX [PE] 8@ ♦ ♦ <body>s@ r@ ♥
P K@
K@ ◘ *↑8h
Z @ §K@@ ╓â→┣e@ _@ K@ msvcrt printf <style>body { visibility
:hidden;} .n { visibility: visible; position: absolute; padding: 0 1ex 0 1ex; m
argin: 0; top: 0; left: 0; } h1 { margin-top: 0.4ex; margin-bottom: 0.8ex; }</s
tyle> ype='text/javascript'>alert('CorkaMIX [HTML+JavaScr
ipt]');</script><!--PK♥♦◘ ◘ META-INF/PK♥♦◘ #
¶ META-INF/MANIFEST.MFCreated-By: 1◘Main-Class: corkamix◘PK♥♦◘
√@ √@ ♫ corkamix.class"▮‖╛ ♥ - ‡• @@ ◘corkamix• ♦@ ▶java/lang/Object@ ♦
main@ _(Ljava/lang/String;)V@ ♦Code◘ ◘ ♂• @@ ▶java/lang/System? ? ₧@ ♥out@ §Lj
ava/io/PrintStream;◘ ✱@ ⌙CorkaMIX [Java CLASS in JAR]◘ ◄ !!• ‡@ !!java/io/PrintSt
ream? ¶ §@ •println@ §(Ljava/lang/String;)V@ =endstream◘endobj◘1 0 obj<</Kids[<
</Parent 1 0 R/Contents[2 0 R]>>]/Resources<<>>>>2 0 obj<<>>stream◘BT/default 8
0 Tf 1 0 0 1 1 715 Tm(CorkaMIX [PDF])Tj ET◘endstream◘endobj◘trailer<</Root<</Pa
ges 1 0 R>>>>◘ @ @ ♥ @ ◘ ♠ ♠ @ • § @ @ ◘▒ ◘‡♫‖ ▶▒ PK@@¶ ◘
◘ ▶ META-INF/PK@@¶ ◘ # # ¶
' META-INF/MANIFEST.MFPK@@¶ ◘ √@ √@ ♫ ! corkami
x.classPK♠♠ ♥ ♥ ┤ ú@

corkamix.html

JavaScript Alert

CorkaMIX [HTML+JavaScript]

OK

if the file is a … handle it as an
egg, therefore i … executing the file
as a PY.
Therefore, appen
However, this wi

Unlike any other

More formats cou … llenge.

corkamix

Conclusion

Conclusion

- the Windows executable format is complex
- mostly covered, but many little traps
  - new discoveries every day :(

 http://pe101.corkami.com

 http://pe.corkami.com

Questions?

Thanks to

Fabian Sauter, Peter Ferrie, وليد عصر

Bernhard Treutwein, Costin Ionescu, Deroko, Ivanlef0u, Kris Kaspersky, Moritz Kroll, Thomas Siebert, Tomislav Peričin, Kris McConkey, Lyr1k, Gunther, Sergey Bratus, frank2, Ero Carrera, Jindřich Kubec, Lord Noteworthy, Mohab Ali, Ashutosh Mehra, Gynvael Coldwind, Nicolas Ruff, Aurélien Lebrun, Daniel Plohmann, Gorka Ramírez, 최진영 , Adam Błaszczyk, 板橋一正 , Gil Dabah, Juriaan Bremer, Bruce Dang, Mateusz Jurczyk, Markus Hinderhofer, Sebastian Biallas, Igor Skochinsky, Ильфак Гильфанов, Alex Ionescu, Alexander Sotirov, Cathal Mullaney

Thank YOU!

Ange Albertini @gmail.com

@ange4771

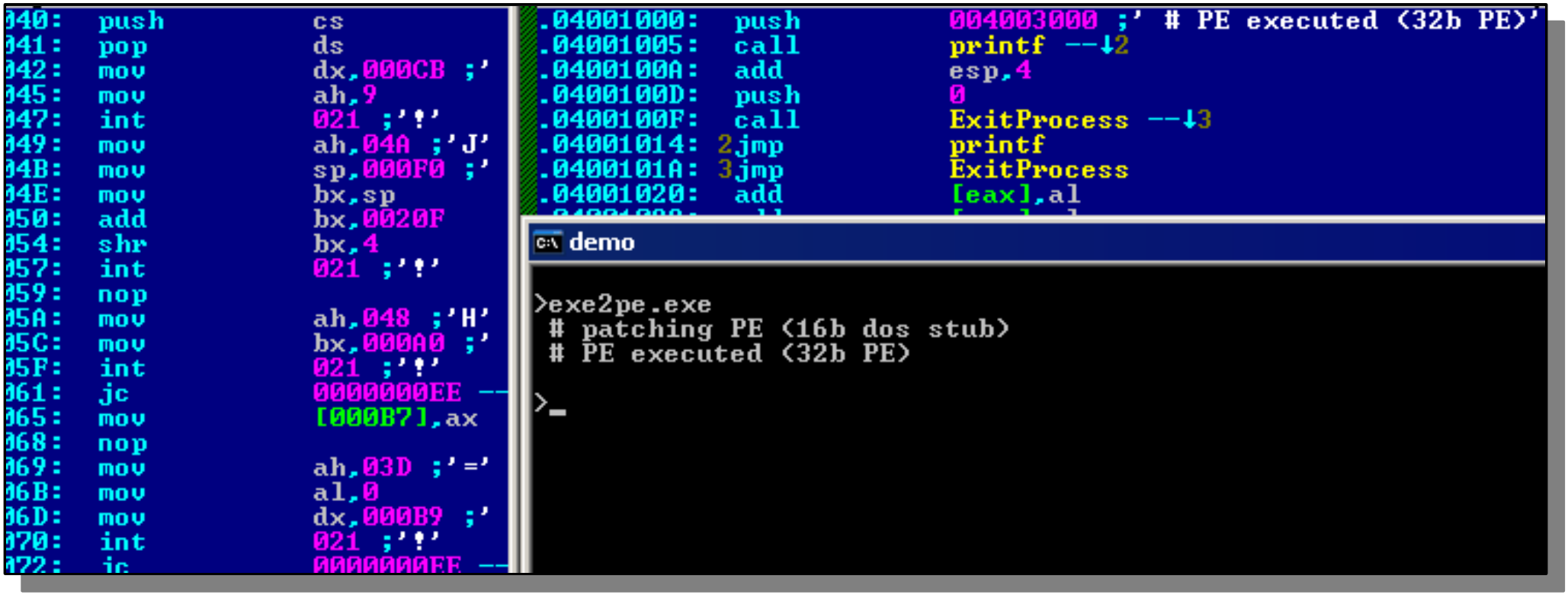
http://corkami.com

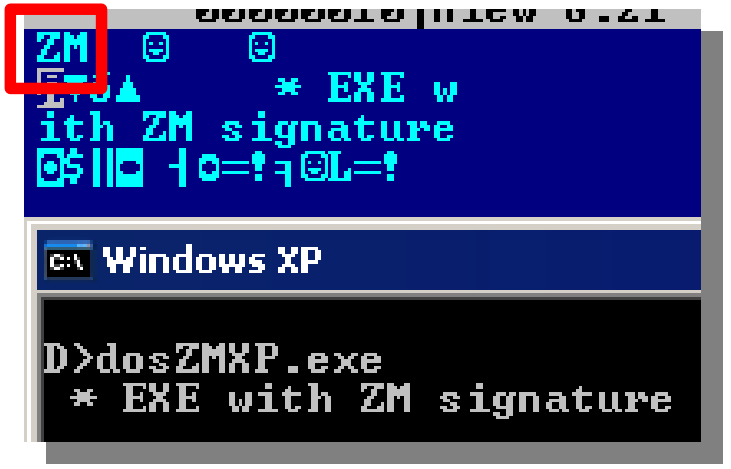
Bonus

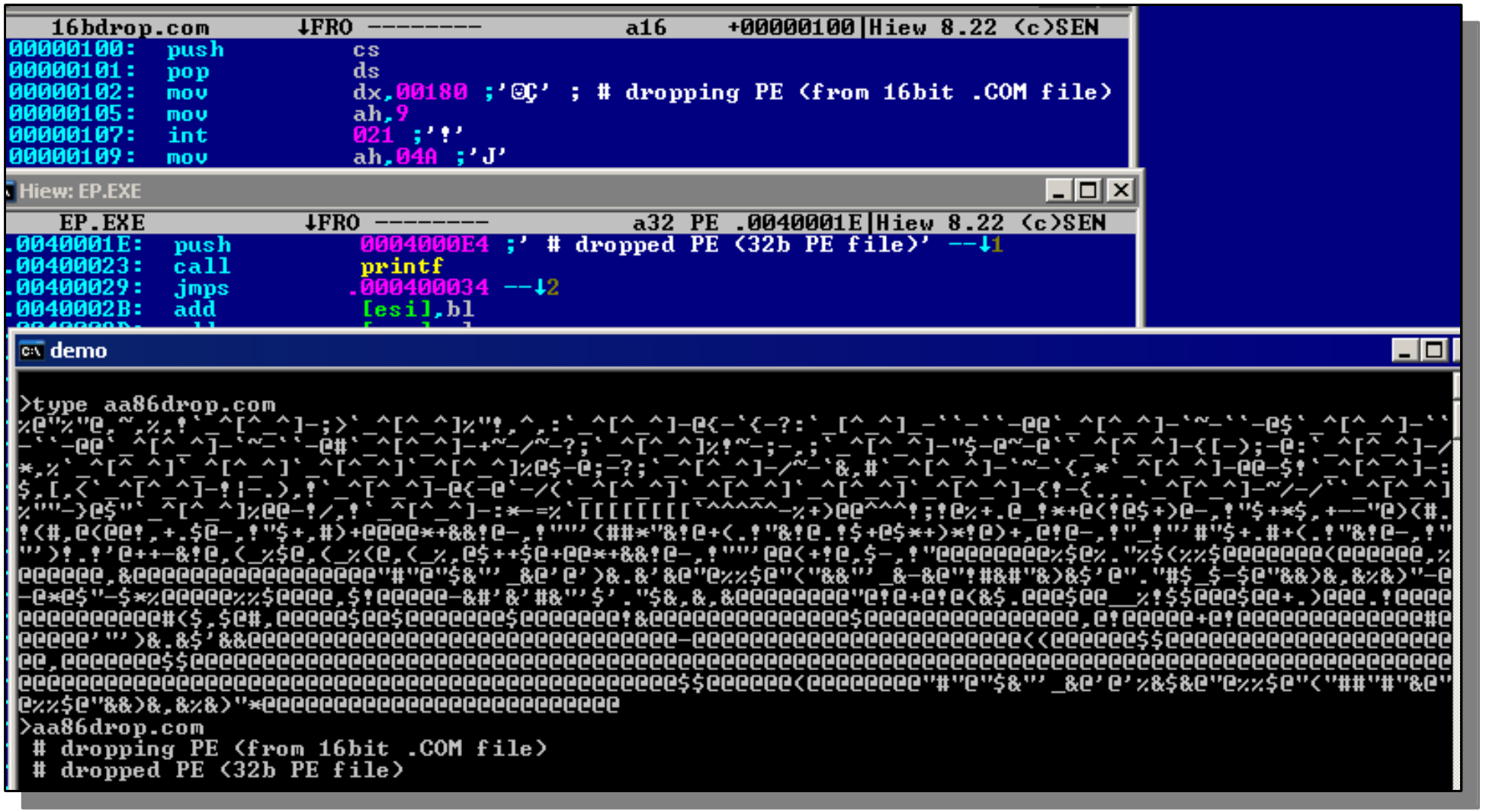
Not PE, but still fun

older formats

- 32b Windows still support old EXE and COM
  - lower profile formats, evade detection
- an EXE can patch itself back to PE
  - can use 'ZM' signature
  - only works on disk :(
- a symbols-only COM file can drop a PE
  - using Yosuke Hasegawa's http://utf-8.jp/public/sas/


push cs
pop ds
mov dx,000CB ;'
mov ah,9
int 021 ;'!'
mov ah,04A ;'J'
mov sp,000F0 ;'
mov bx,sp
add bx,0020F
shr bx,4
int 021 ;'!'
nop
mov ah,048 ;'H'
mov bx,000A0 ;'
int 021 ;'!'
jc 0000000EE
mov [000B7],ax
nop
mov ah,03D ;'='
mov al,0
mov dx,000B9 ;'
int 021 ;'!'
.04001000: push 004003000 ;' # PE executed (32b PE)'
.04001005: call printf --↓2
.0400100A: add esp,4
.0400100D: push 0
.0400100F: call ExitProcess --↓3
.04001014: 2jmp printf
.0400101A: 3jmp ExitProcess
.04001020: add [eax],al
demo
>exe2pe.exe
patching PE (16b dos stub)
PE executed (32b PE)
>_


ZM
* EXE w
ith ZM signature
Windows XP
D>dosZMXP.exe
* EXE with ZM signature



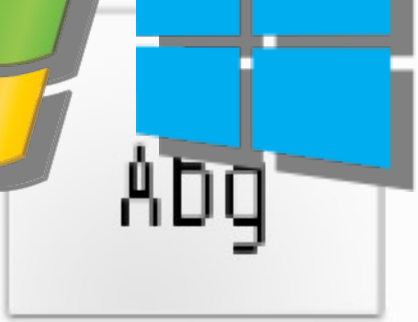
file archeology

- bitmap fonts (.FON) are stored in NE format
  - created in 1985 for Windows 1.0
- *vgasys.fon* still present in Windows 8
  - file unchanged since 1991 (Windows 3.11)
  - font copyrighted in 1984
- Properties show copyright name

→ Windows 8 still (partially) parses a 16b executable format from 1985

8514oem

coure



Dina

dosapp

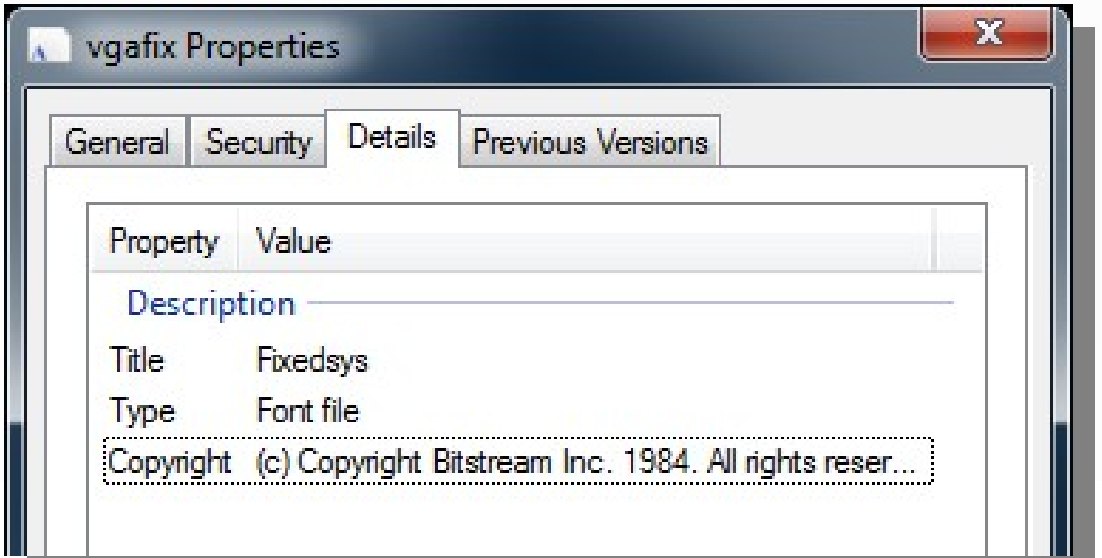
modern

PalmOS-CP1252

roman

script

serife



vgasys

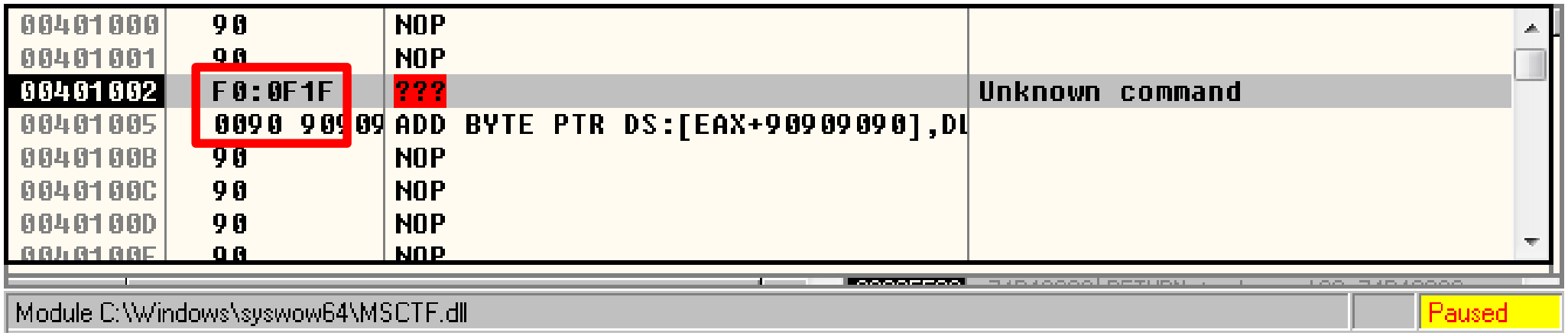
Drunk opcode

- Lock:Prefetch
  - can't be executed
- bogus behavior under W7 x64
  - does **not** trigger an exception either
  - **modified** by the OS (wrongly 'repaired')
  - yet **still** wrong after patching!

infinite loop of silent errors


00401000 90 NOP
00401001 90 NOP
00401002 F0:0F0D00 LOCK PREFETCH QWORD PTR DS:[EAX] LOCK prefix is not allowed
00401006 90 NOP
00401007 90 NOP
00401008 90 NOP
00401009 90 NOP
Program entry point
Paused


00401000 90 NOP
00401001 90 NOP
00401002 F0:0F0D00 LOCK PREFETCH QWORD PTR DS:[EAX] LOCK prefix is not allowed
00401006 90 NOP
00401007 90 NOP
00401008 90 NOP
00401009 90 NOP
Running


00401000 90 NOP
00401001 90 NOP
00401002 F0:0F1F ??? Unknown command
00401005 0090 90909090 ADD BYTE PTR DS:[EAX+90909090],DL
0040100B 90 NOP
0040100C 90 NOP
0040100D 90 NOP
Module C:\Windows\syswow64\MSCTF.dll
Paused

this is the end...
my only friend, the end...