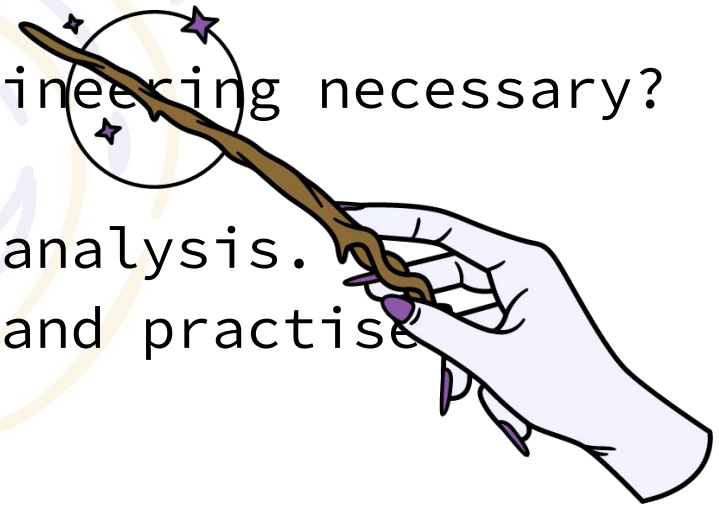
The Sorcery of Malware Reverse Engineering

Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise



Contents

What is Malware ?

Malware is a code that is used to perform malicious actions.

Types of Malwares

- ★ Virus
- ★ Worm
- ★ Trojan
- ★ Spyware
- ★ Ransomware
- ★ Adware
- ★ Rootkit

...

Types of Malwares

★ Virus

- ❏ Replicating ability.

- ❏ Delete
- ❏ Encrypt Files
- ❏ Modify Applications
- ❏ Disable functions

Types of Malwares

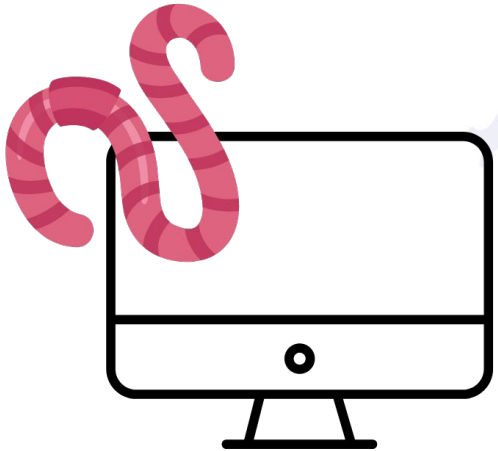
Key points:

- ★ Attached to files to corrupt them.
- ★ Multiply.
- ★ Requires an existing file to infect.
- ★ Needs a person to initiate itself that is, not self propagating

Types of Malwares

★ Worm

Self-replicate and propagate independently.
Stand-alone malicious programs.



- ❏ Enters via the network or downloaded file
- ❏ Multiply and spread

Types of Malwares

Key points:

- ★ Replicate themselves.
- ★ No need of activation.
- ★ Multiply and send it self without user intervention.
- ★ Fast propagation.

Types of Malwares

★ Trojan

Type of malicious code or software that looks legitimate but can take control of your computer.

Types of Malwares

Key points:

- ★ Pretends to be legitimate.
- ★ Deceive user and get into system.
- ★ Cannot replicate.

Types of Malwares

★ Spyware

Software that infiltrates your computing device, stealing your sensitive information.

- ❏ Bank account information.
- ❏ Steal your personal identity.

Types of Malwares

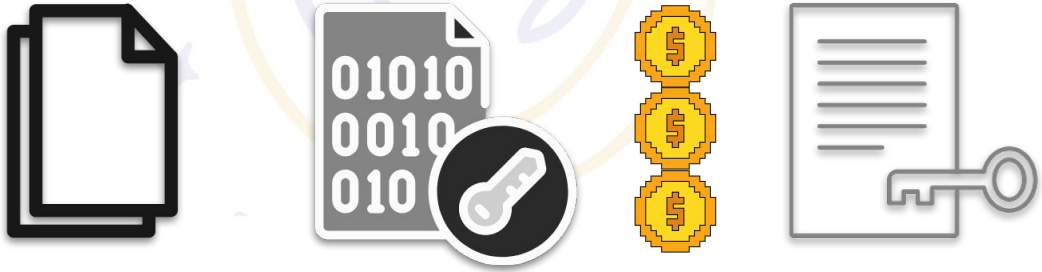
Key points:

- ★ Installed without the user’s knowledge.
- ★ Gathers information about the user.
- ★ Thus spys on the user and his activities.
- ★ Keylogging

Types of Malwares

★ Ransomware

A form of malware that encrypts victim's files and data.



Types of Malwares

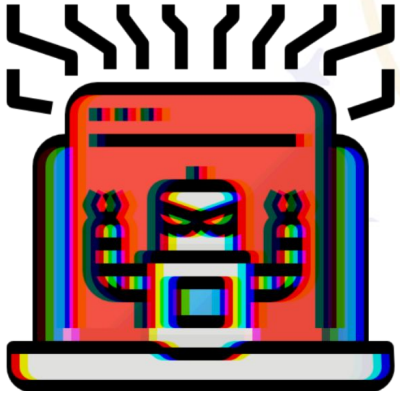
Key points:

★ Encrypts your files and in return demands for money for the decryption key.

Types of Malwares

★ Rootkit

Malicious software that allows an unauthorized user to have privileged access.



Types of Malwares

Key points:

- ★ Hide themselves within another application and uses operating system’s file to hide itself, such as registry key, running processes etc.
- ★ Hard to find and clean-out.


Check Examples

Samples

- WANNACRY
- ILOVEYOU
- STUXNET
- Cryptolocker
- Petya
- Zeus
- ...

Sample - 1

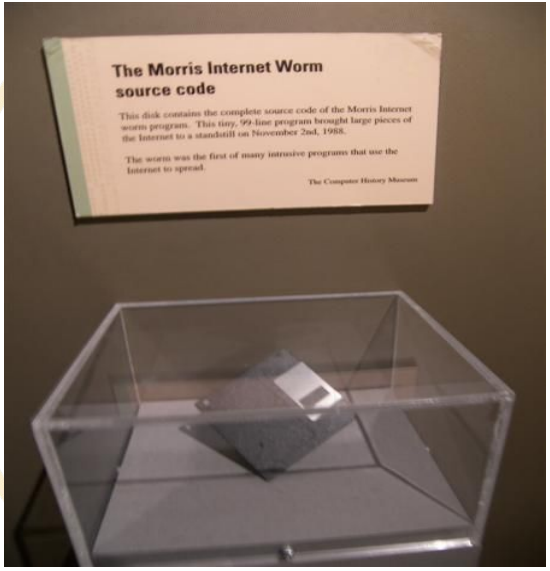
★ Morris worm

A form of malware that encrypts victim's files and data.

- ❏ November 1988
- ❏ How big the internet was?
- ❏ Worked well
  - ❏ Copying
  - ❏ Send signals
- ❏ First DDos attack

Sample - 1

★ Morris worm
“99 line program”



https://en.wikipedia.org/wiki/Morris_worm

Sample -II

★ ILOVEYOU

- ❑ WORM
- ❑ 2000
- ❑ 10 MILLION Windows PCs

Sample -III

★ Wannacry

- ❑ Ransomware
- ❑ May 2017
- ❑ 200,000 devices

Sample -IV

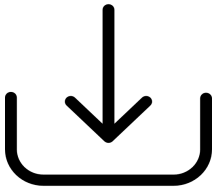
★ Zeus

- ❏ Trojan-horse
- ❏ July 2007
- ❏ By 2009, it compromised many accounts present on the companies:

https://en.wikipedia.org/wiki/Zeus_(malware)

Sample -IV

- Also came to be known as zbot.
- Creates a botnet.

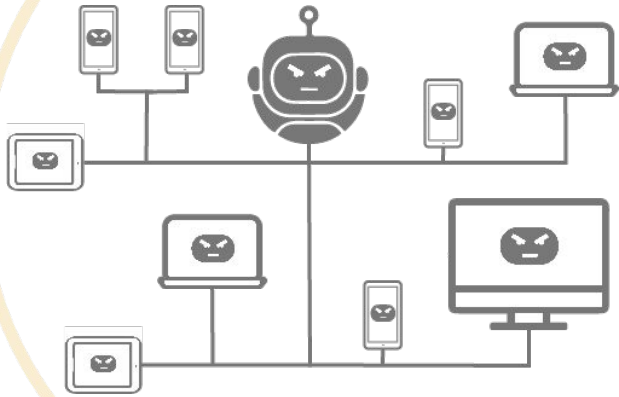


https://en.wikipedia.org/wiki/Zeus_(malware)

Types of Malwares

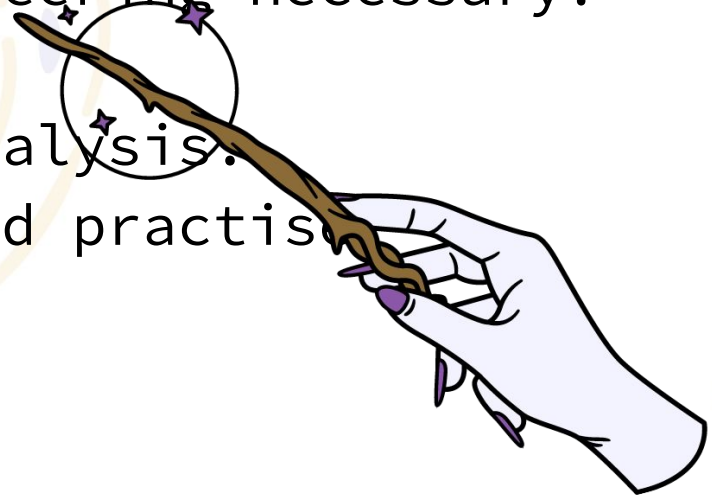
★ Botnet

Networks of computers infected by malware (such as viruses, keyloggers and other malicious software) and controlled remotely by attackers.



Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.



Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
  - Dealing with an uninvited guest on your system.
  - Prevention and cure for the infected systems.
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.

Scenario

Suppose that a system is infected with a malware...

- ❏ Collect the evidences:
  - ❏ File
  - ❏ Network traffic analysis.
  - ❏ Changes in the system.
  - ❏ ...

Why is reverse engineering necessary ?

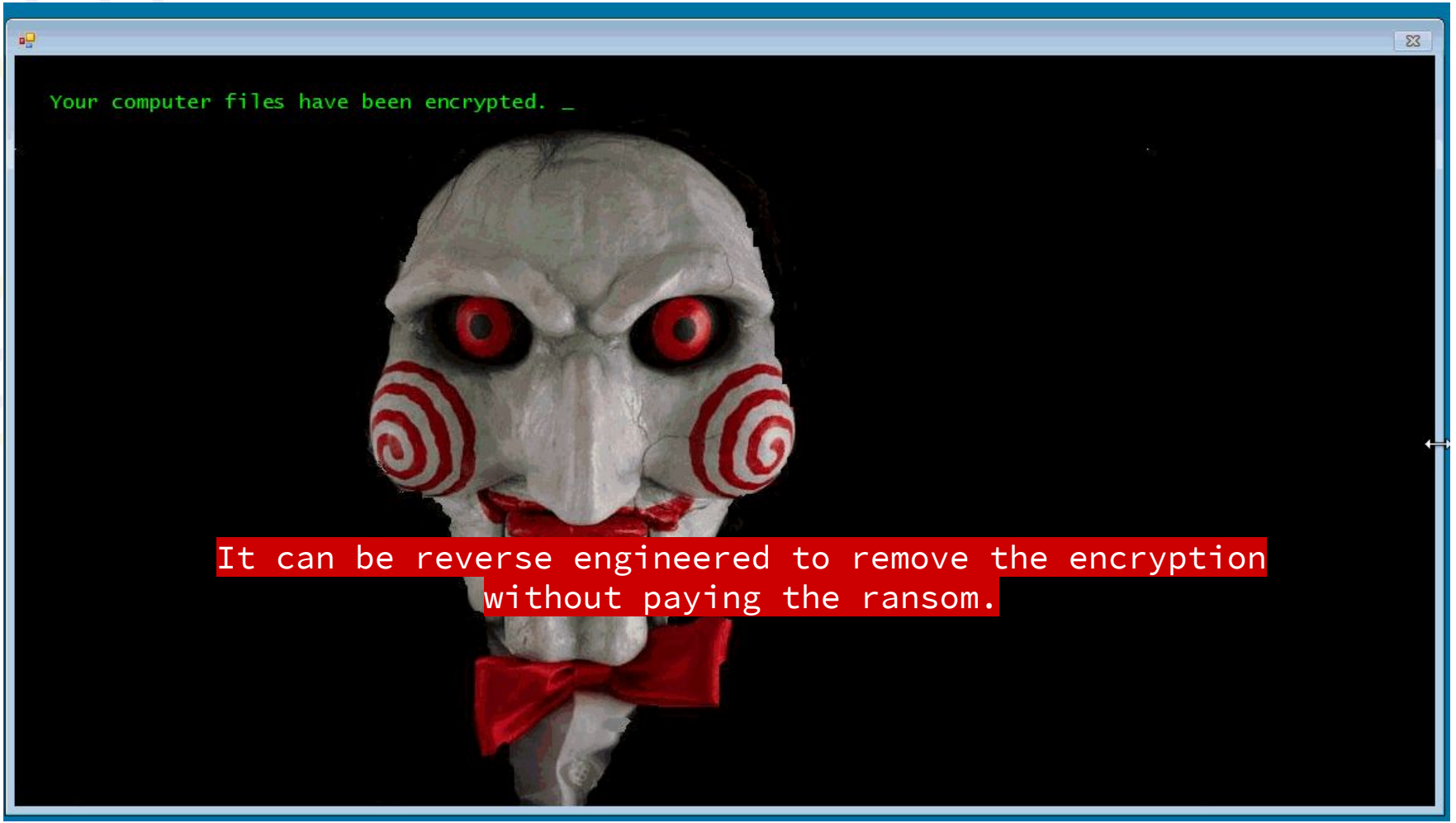
- Backtrack from the current information.
  - Reverse the logic.
  - Backtrack from the current information.
  - Search the sources.

Sample -V

★ Jigsaw

- ❏ Ransomware
- ❏ 2016
- ❏ Written in .NET Framework

https://en.wikipedia.org/wiki/Jigsaw_(ransomware)


Your computer files have been encrypted. _
It can be reverse engineered to remove the encryption without paying the ransom.

Precaution

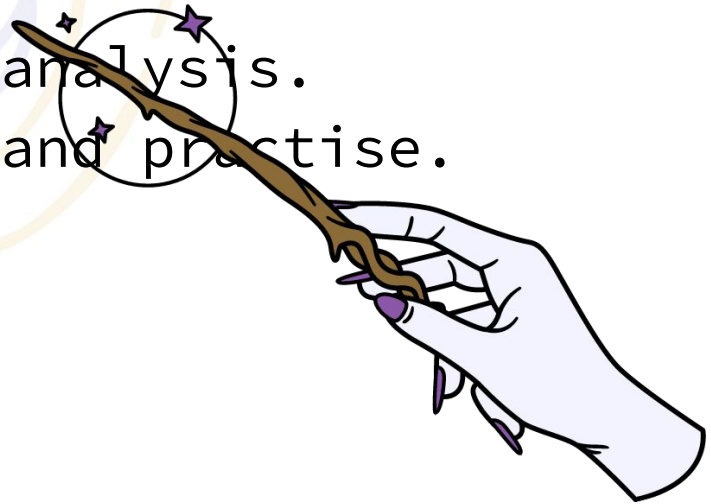
- Install Anti-Virus/Malware Software.
- Keep Your Anti-Virus Software up to Date.
- Keep your softwares/applications updated.
- Back-up your data.
- Never download attachments from unknown sources.
- Limit personal information you give online.

Precaution:

- ❏ Enable multi-factor authentication
- ❏ Secure ports and services that are exposed on the internet
- ❏ Don’t click on untrusted email attachments or web links
- ❏ Download software and contents from trusted sources only.
- ❏ Clean out outdated and unused accounts.

Contents

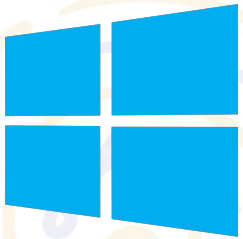
- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.



Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
  - PE file structure.
  - Why is it necessary to know.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.

PE File Structure



Microsoft Windows

DOS -> NE -> LX -> PE

DISK OPERATING SYSTEM

New Executable

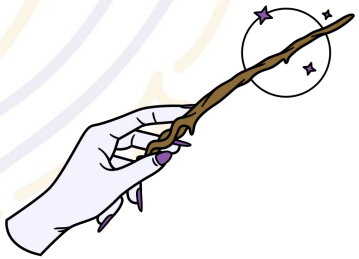
Linear Executable

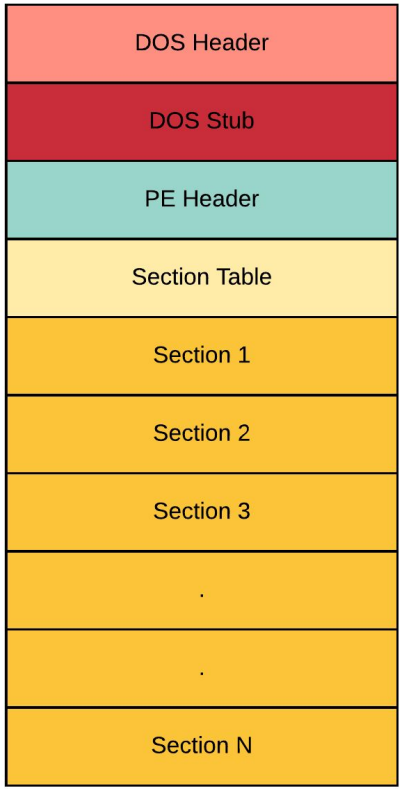
Portable Executable

PE File Structure

- ★ PE: Portable executable
  - ❏ Executables
  - ❏ DLL
  - ❏ .NET
  - ❏ ...
- ★ Tools
  - ❏ CFF explorer
  - ❏ PE bear

PE File Structure

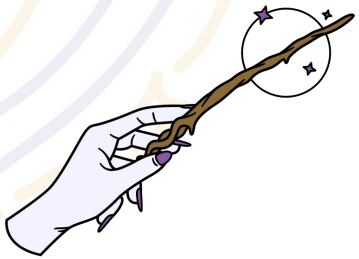




DOS headers

- Backward compatibility to MS-DOS, to run and exit without any errors.
- If the program runs under MS-DOS then it will run and print the message "This program cannot be run in DOS mode" and exit without any errors.
- This is the first 64 BYTES of the file.

PE File Structure



DOS Header
DOS Stub
PE Header
Section Table
Section 1
Section 2
Section 3
.
.
Section N

DOS headers

DOS header members:

```
e_magic
e_cblp
e_cp
e_crlc
e_cparhdr
e_minalloc
e_maxalloc
e_ss
e_sp
e_csum
e_ip
e_ip
e_ifarlc
e_ovna
e_res
e_oemid
e_oeminfo
e_res2
e_lfanew
```

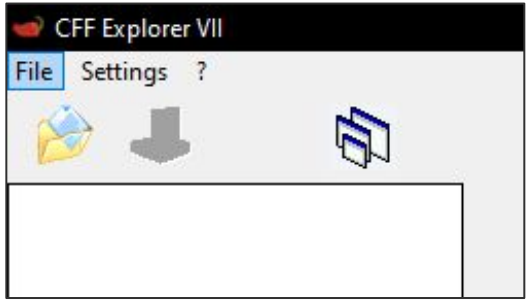

Practice Time !!

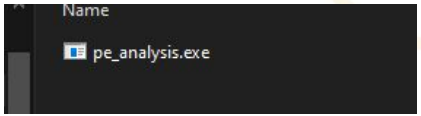
DOS headers

```
C:\Users\hp\Desktop\workshoptemp>pe_analysis.exe
Welcome to this workshop!!
C:\Users\hp\Desktop\workshoptemp>_
```

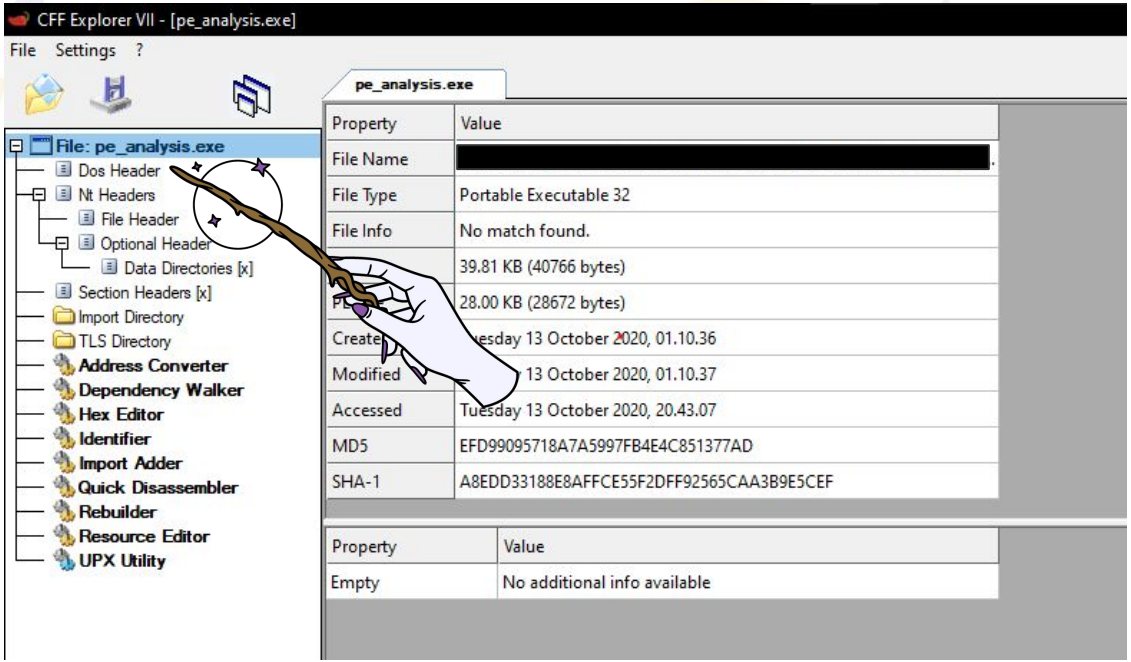
DOS headers


CFF Explorer
App

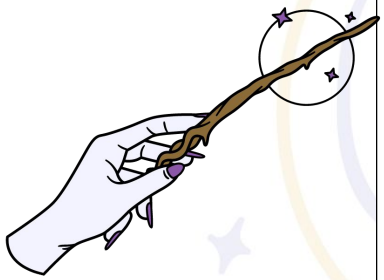

CFF Explorer VII
File
Settings
?


Name
pe_analysis.exe

DOS headers


CFF Explorer VII - [pe_analysis.exe]
File Settings ?
pe_analysis.exe
File: pe_analysis.exe
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
Import Directory
TLS Directory
Address Converter
Dependency Walker
Hex Editor
Identifier
Import Adder
Quick Disassembler
Rebuilder
Resource Editor
UPX Utility
Property
Value
File Name
File Type
Portable Executable 32
File Info
No match found.
39.81 KB (40766 bytes)
28.00 KB (28672 bytes)
Modified
Accessed
Tuesday 13 October 2020, 20.43.07
MD5
EFD99095718A7A5997FB4E4C851377AD
SHA-1
A8EDD33188E8AFFCE55F2DFF92565CAA3B9E5CEF
Property
Value
Empty
No additional info available

DOS headers



CFF Explorer VII - [pe_analysis.exe]

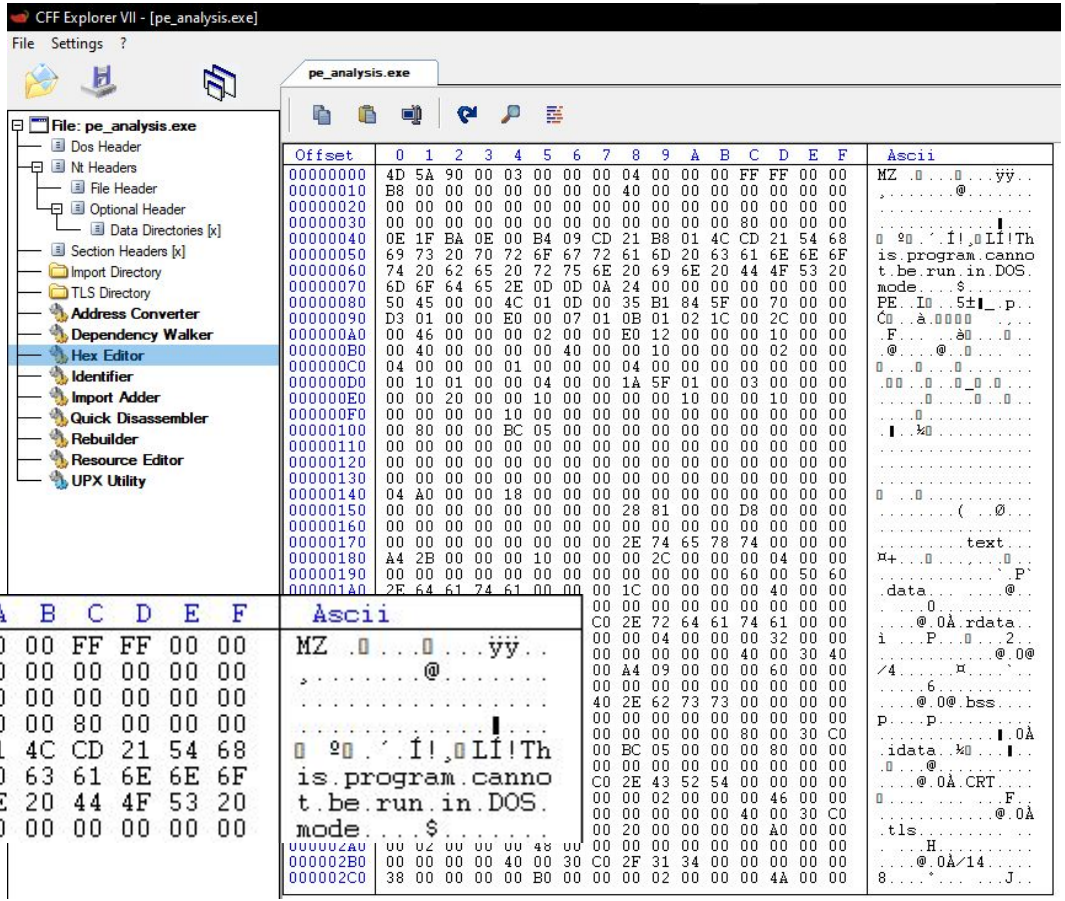
File Settings ?

pe_analysis.exe

- File: pe_analysis.exe
  - Dos Header
  - Nt Headers
    - File Header
    - Optional Header
      - Data Directories [x]
  - Section Headers [x]
  - Import Directory
  - TLS Directory
  - **Address Converter**
  - **Dependency Walker**
  - **Hex Editor**
  - **Identifier**
  - **Import Adder**
  - **Quick Disassembler**
  - **Rebuilder**
  - **Resource Editor**
  - **UPX Utility**

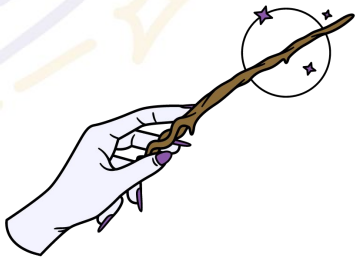
Member	Offset	Size	Value
e_minalloc	0000000A	Word	0000
e_maxalloc	0000000C	Word	FFFF
e_ss	0000000E	Word	0000
e_sp	00000010	Word	00B8
e_csum	00000012	Word	0000
e_ip	00000014	Word	0000
e_cs	00000016	Word	0000
e_lfarlc	00000018	Word	0040
e_ovno	0000001A	Word	0000
e_res	0000001C	Word	0000
	0000001E	Word	0000
	00000020	Word	0000
	00000022	Word	0000
e_oemid	00000024	Word	0000
e_oeminfo	00000026	Word	0000
e_res2	00000028	Word	0000
	0000002A	Word	0000
	0000002C	Word	0000
	0000002E	Word	0000
	00000030	Word	0000
	00000032	Word	0000
	00000034	Word	0000
	00000036	Word	0000
	00000038	Word	0000
	0000003A	Word	0000
e_lfanew	0000003C	Dword	00000080

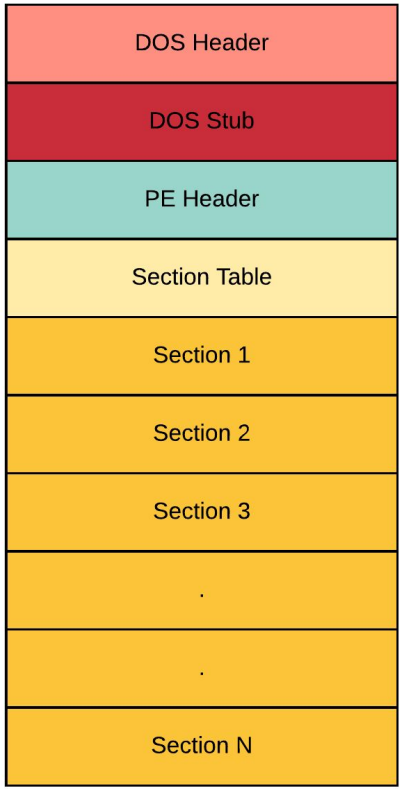
DOS headers



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ ..... ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00	............€...
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....$.......

PE File Structure



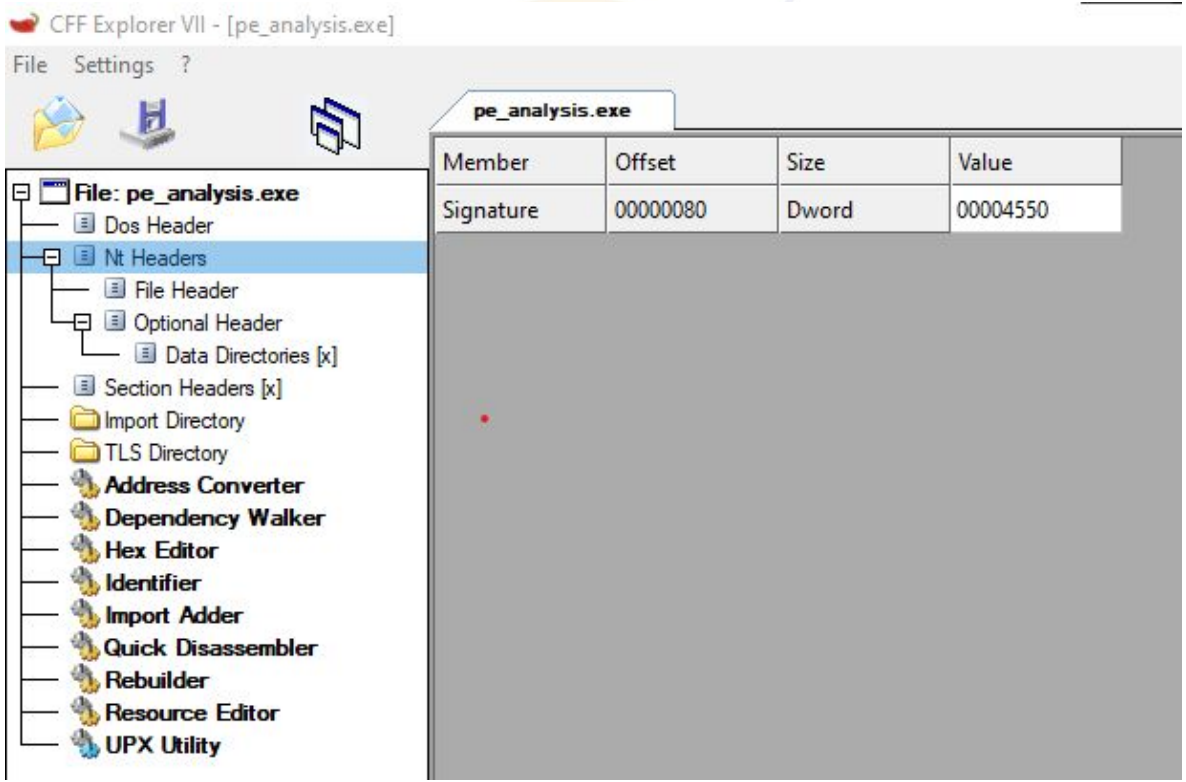


PE headers

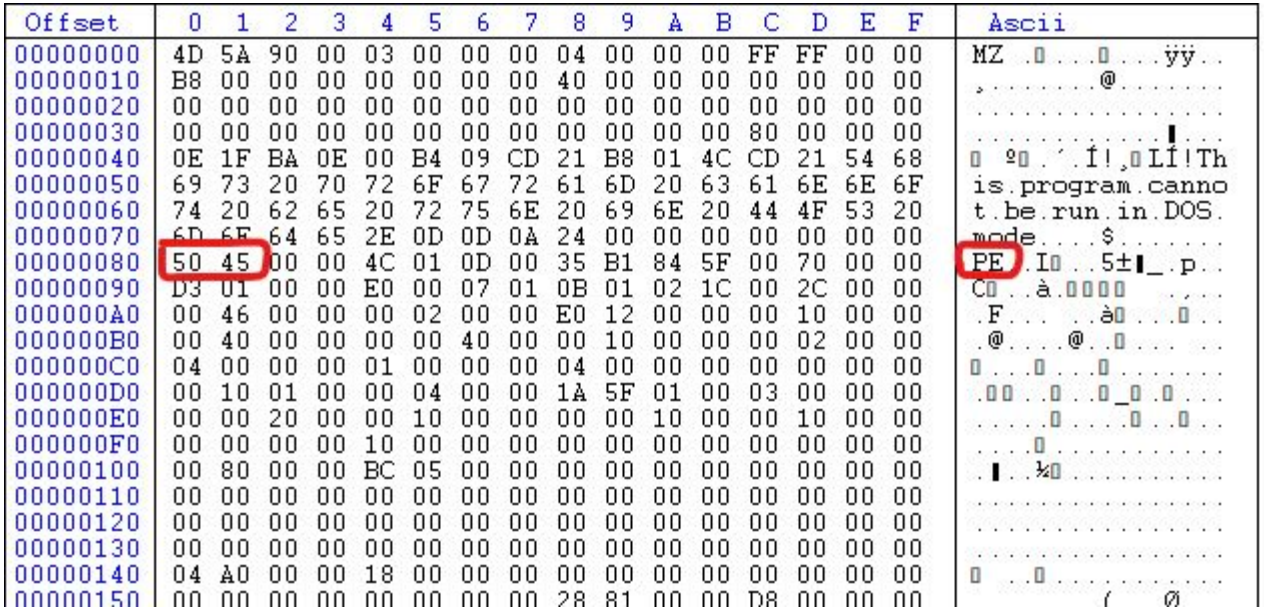
- Starts with the value
- 50h, 45h, 00h, 00h
- ("PE" followed by two terminating zeroes).
- It is a struct with the following members:

```
IMAGE_NT_HEADERS STRUCT {
      Signature
      FileHeader
      OptionalHeader
}
```

PE headers


CFF Explorer VII - [pe_analysis.exe]
File
Settings
?
pe_analysis.exe
File: pe_analysis.exe
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
Import Directory
TLS Directory
Address Converter
Dependency Walker
Hex Editor
Identifier
Import Adder
Quick Disassembler
Rebuilder
Resource Editor
UPX Utility
Member
Offset
Size
Value
Signature
00000080
Dword
00004550

PE headers

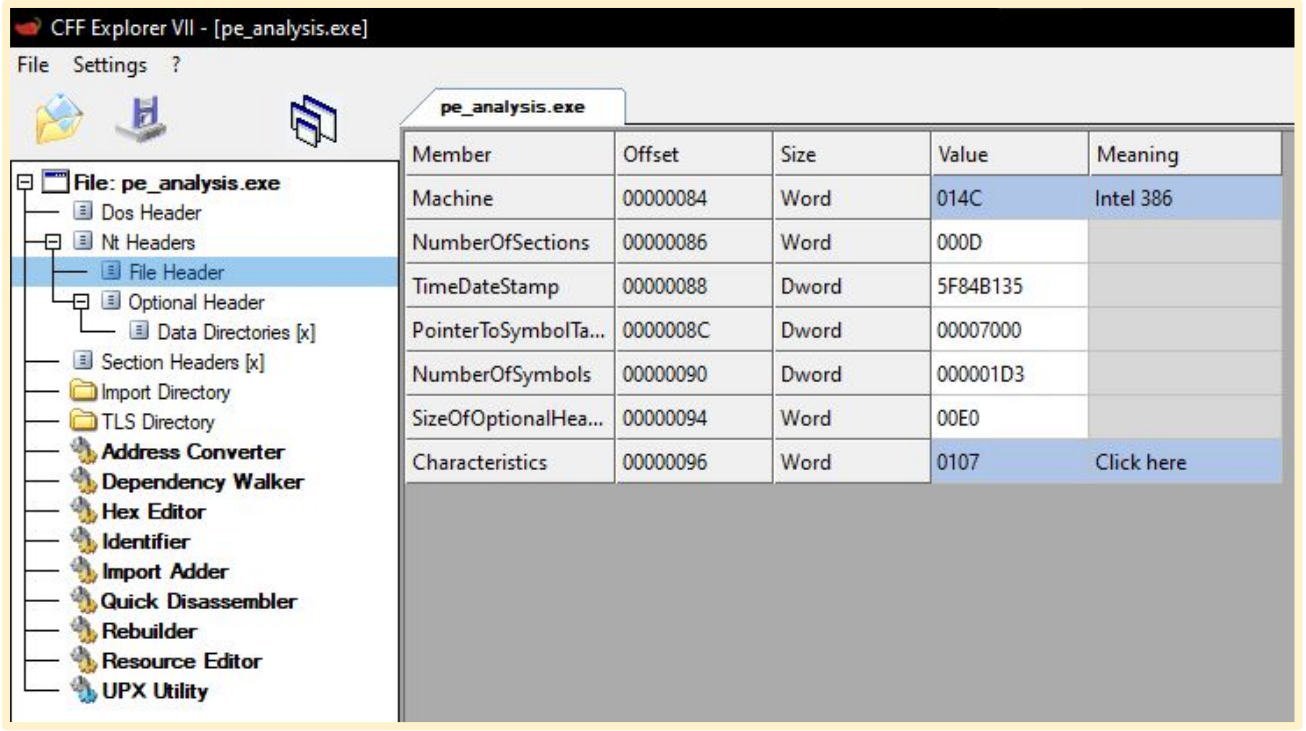


Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ .□ ...□ ...ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.......
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00	............▌...
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....$.......
00000080	50	45	00	00	4C	01	0D	00	35	B1	84	5F	00	70	00	00	PE..L□..5±▌_.p..
00000090	D3	01	00	00	E0	00	07	01	0B	01	02	1C	00	2C	00	00	Ó□..à.□□□□ ..,..
000000A0	00	46	00	00	00	02	00	00	E0	12	00	00	00	10	00	00	.F.....à□...□..
000000B0	00	40	00	00	00	00	40	00	00	10	00	00	00	02	00	00	.@....@..□.....
000000C0	04	00	00	00	01	00	00	00	04	00	00	00	00	00	00	00	□...□...□.......
000000D0	00	10	01	00	00	04	00	00	1A	5F	01	00	03	00	00	00	.□□..□..□_□.□...
000000E0	00	00	20	00	00	10	00	00	00	00	10	00	00	10	00	00	.....□....□..□..
000000F0	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	....□...........
00000100	00	80	00	00	BC	05	00	00	00	00	00	00	00	00	00	00	.▌..¼□..........
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000140	04	A0	00	00	18	00	00	00	00	00	00	00	00	00	00	00	□ ..□...........
00000150	00	00	00	00	00	00	00	00	28	81	00	00	D8	00	00	00	(Ø

PE header : File Header

```
IMAGE_FILE_HEADER_STRUCT{
Machine
NumberOfSections
TimeDateStamp
PointerToSymbolTable
NumberOfSymbols
SizeOfOptionalHeader
Characteristics
}
```

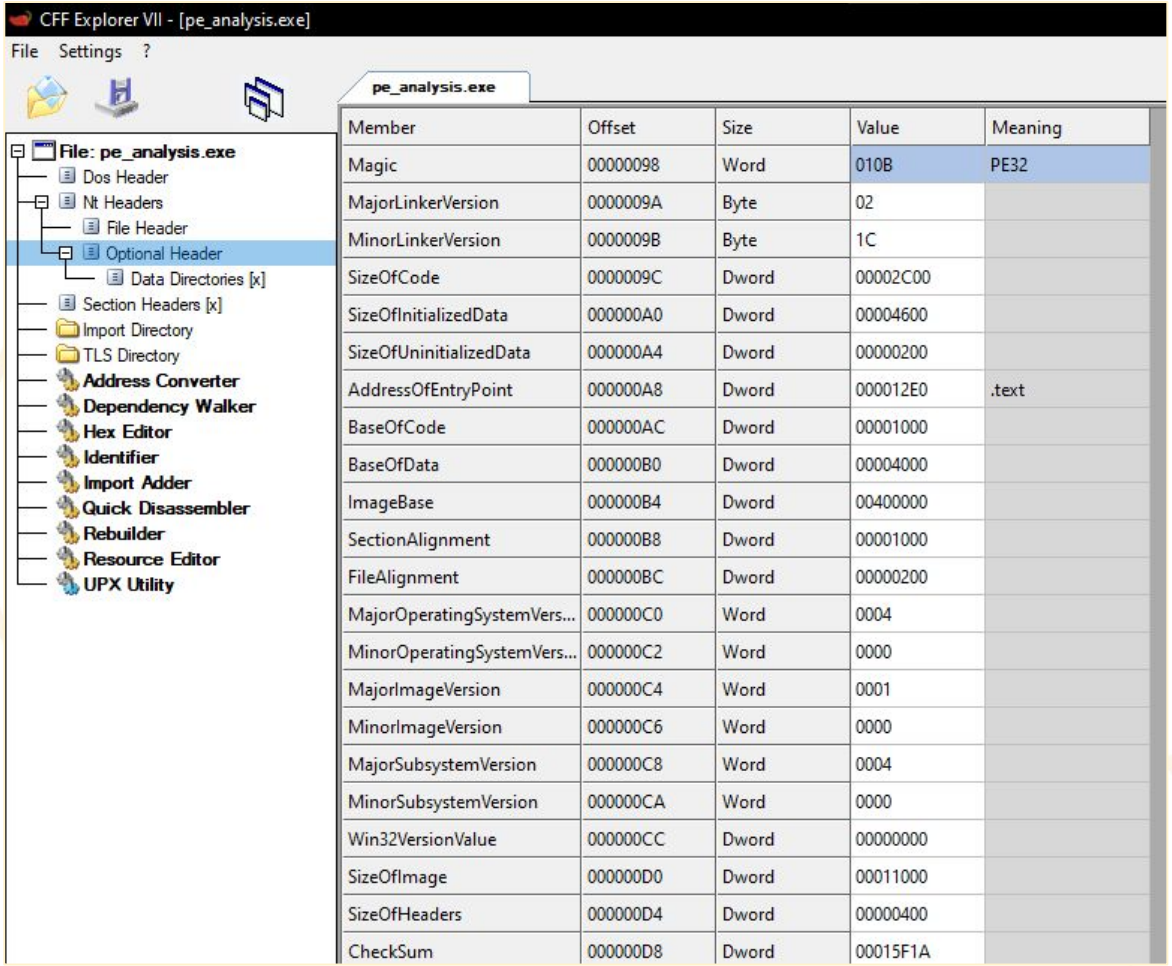
PE header : File Header



PE header : Optional Headers

```
IMAGE_OPTIONAL_HEADER_STRUCT {
  Magic
  MajorLinkerVersion
  MinorLinkerVersion
  SizeOfCode
  SizeOfInitializedData
  SizeOfUninitializedData
  AddressOfEntryPoint
  BaseOfCode
  BaseOfData
  ImageBase
  SectionAlignment
  FileAlignment
  MajorOperatingSystemVersion
  MinorOperatingSystemVersion
  MajorImageVersion
  MinorImageVersion
  MajorSubsystemVersion
  MinorSubsystemVersion
  Win32VersionValue
  SizeOfImage
  SizeOfHeaders
  CheckSum
  Subsystem
  DllCharacteristics
  SizeOfStackReserve
  SizeOfStackCommit
  SizeOfHeapReserve
  SizeOfHeapCommit
  LoaderFlags
  NumberOfRvaAndSizes
  DataDirectory
}
```

PE header : Optional Headers



Member	Offset	Size	Value	Meaning
Magic	00000098	Word	010B	PE32
MajorLinkerVersion	0000009A	Byte	02	
MinorLinkerVersion	0000009B	Byte	1C	
SizeOfCode	0000009C	Dword	00002C00	
SizeOfInitializedData	000000A0	Dword	00004600	
SizeOfUninitializedData	000000A4	Dword	00000200	
AddressOfEntryPoint	000000A8	Dword	000012E0	.text
BaseOfCode	000000AC	Dword	00001000	
BaseOfData	000000B0	Dword	00004000	
ImageBase	000000B4	Dword	00400000	
SectionAlignment	000000B8	Dword	00001000	
FileAlignment	000000BC	Dword	00000200	
MajorOperatingSystemVers...	000000C0	Word	0004	
MinorOperatingSystemVers...	000000C2	Word	0000	
MajorImageVersion	000000C4	Word	0001	
MinorImageVersion	000000C6	Word	0000	
MajorSubsystemVersion	000000C8	Word	0004	
MinorSubsystemVersion	000000CA	Word	0000	
Win32VersionValue	000000CC	Dword	00000000	
SizeOfImage	000000D0	Dword	00011000	
SizeOfHeaders	000000D4	Dword	00000400	
CheckSum	000000D8	Dword	00015F1A	

PE header : Optional Headers

DataDirectory

- 128 bytes
- Array of `IMAGE_DATA_DIRECTORY` structure

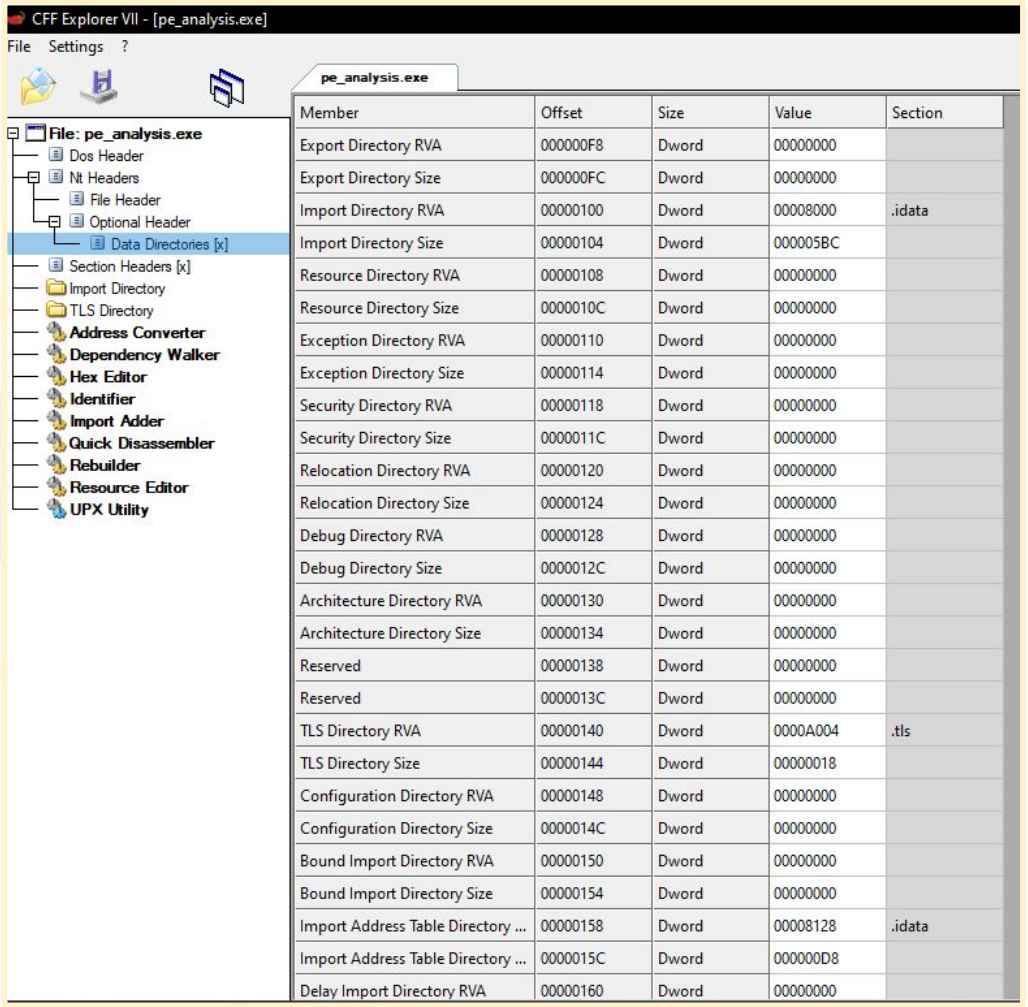
```
IMAGE_DATA_DIRECTORY_STRUCT {
        DWORD VirtualAddress
        DWORD Size
    }
```

PE header : Optional Headers

Optional headers

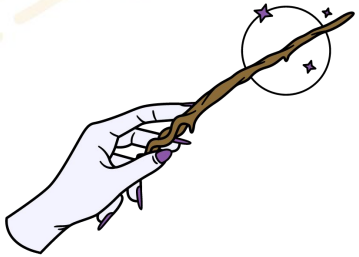
```
IMAGE_DIRECTORY_ENTRY_EXPORT
IMAGE_DIRECTORY_ENTRY_IMPORT
IMAGE_DIRECTORY_ENTRY_RESOURCE
IMAGE_DIRECTORY_ENTRY_EXCEPTION
IMAGE_DIRECTORY_ENTRY_SECURITY
IMAGE_DIRECTORY_ENTRY_BASERELOC
IMAGE_DIRECTORY_ENTRY_DEBUG
IMAGE_DIRECTORY_ENTRY_COPYRIGHT
IMAGE_DIRECTORY_ENTRY_GLOBALPTR
IMAGE_DIRECTORY_ENTRY_TLS
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT
IMAGE_DIRECTORY_ENTRY_IAT
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR
IMAGE_NUMBEROF_DIRECTORY_ENTRIES
```

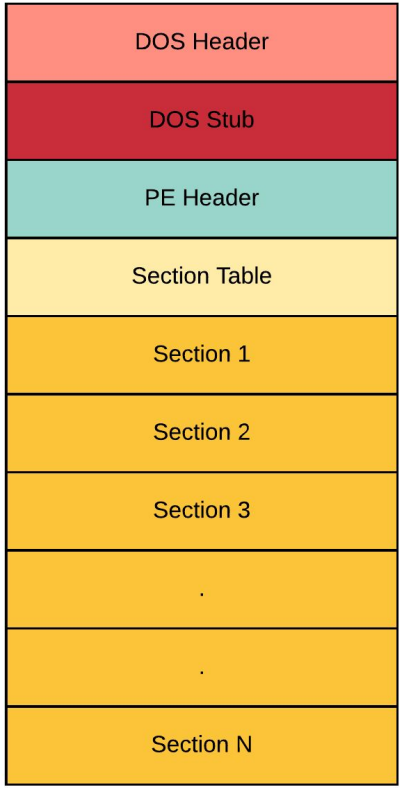
PE header : Optional Headers



Member	Offset	Size	Value	Section
Export Directory RVA	000000F8	Dword	00000000	
Export Directory Size	000000FC	Dword	00000000	
Import Directory RVA	00000100	Dword	00008000	.idata
Import Directory Size	00000104	Dword	000005BC	
Resource Directory RVA	00000108	Dword	00000000	
Resource Directory Size	0000010C	Dword	00000000	
Exception Directory RVA	00000110	Dword	00000000	
Exception Directory Size	00000114	Dword	00000000	
Security Directory RVA	00000118	Dword	00000000	
Security Directory Size	0000011C	Dword	00000000	
Relocation Directory RVA	00000120	Dword	00000000	
Relocation Directory Size	00000124	Dword	00000000	
Debug Directory RVA	00000128	Dword	00000000	
Debug Directory Size	0000012C	Dword	00000000	
Architecture Directory RVA	00000130	Dword	00000000	
Architecture Directory Size	00000134	Dword	00000000	
Reserved	00000138	Dword	00000000	
Reserved	0000013C	Dword	00000000	
TLS Directory RVA	00000140	Dword	0000A004	.tls
TLS Directory Size	00000144	Dword	00000018	
Configuration Directory RVA	00000148	Dword	00000000	
Configuration Directory Size	0000014C	Dword	00000000	
Bound Import Directory RVA	00000150	Dword	00000000	
Bound Import Directory Size	00000154	Dword	00000000	
Import Address Table Directory ...	00000158	Dword	00008128	.idata
Import Address Table Directory ...	0000015C	Dword	000000D8	
Delay Import Directory RVA	00000160	Dword	00000000	

PE File Structure

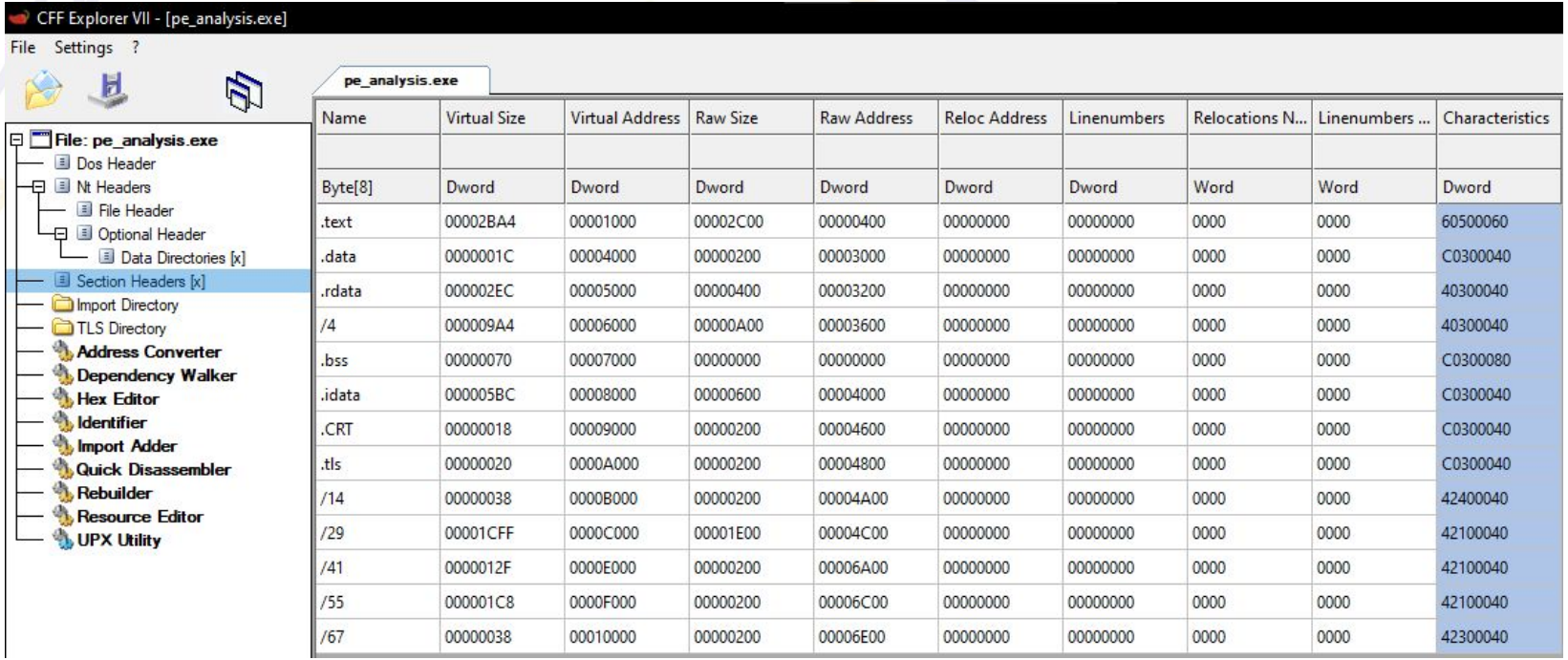




Section Table

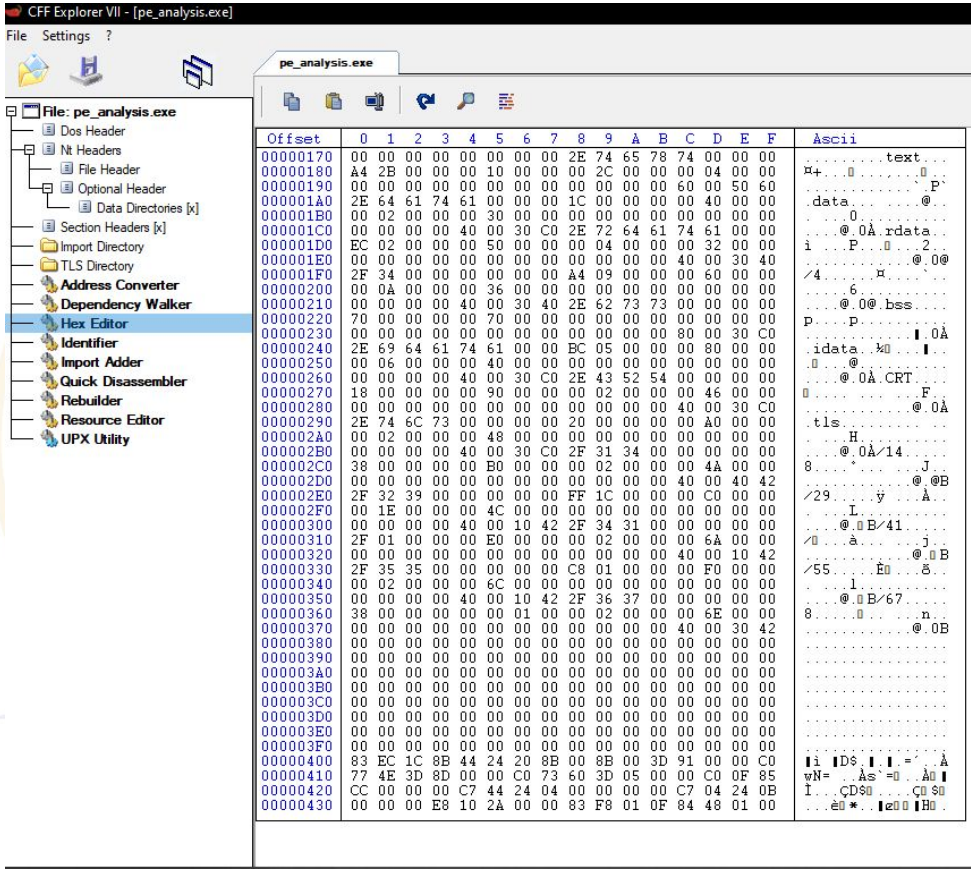
- The number of the array members is determined by NumberOfSections field in the file header struct or IMAGE_FILE_HEADER
- Each Section header
  - Size is at least 40 bytes.
  - Contains the following informations:
    - Name
    - PhysicalAddress
    - VirtualAddress
    - SizeOfRawData
    - PointerToRawData
    - Characteristics

Section Headers



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00002BA4	00001000	00002C00	00000400	00000000	00000000	0000	0000	60500060
.data	0000001C	00004000	00000200	00003000	00000000	00000000	0000	0000	C0300040
.rdata	000002EC	00005000	00000400	00003200	00000000	00000000	0000	0000	40300040
/4	000009A4	00006000	00000A00	00003600	00000000	00000000	0000	0000	40300040
.bss	00000070	00007000	00000000	00000000	00000000	00000000	0000	0000	C0300080
.idata	000005BC	00008000	00000600	00004000	00000000	00000000	0000	0000	C0300040
.CRT	00000018	00009000	00000200	00004600	00000000	00000000	0000	0000	C0300040
.tls	00000020	0000A000	00000200	00004800	00000000	00000000	0000	0000	C0300040
/14	00000038	0000B000	00000200	00004A00	00000000	00000000	0000	0000	42400040
/29	00001CFF	0000C000	00001E00	00004C00	00000000	00000000	0000	0000	42100040
/41	0000012F	0000E000	00000200	00006A00	00000000	00000000	0000	0000	42100040
/55	000001C8	0000F000	00000200	00006C00	00000000	00000000	0000	0000	42100040
/67	00000038	00010000	00000200	00006E00	00000000	00000000	0000	0000	42300040

Section Headers

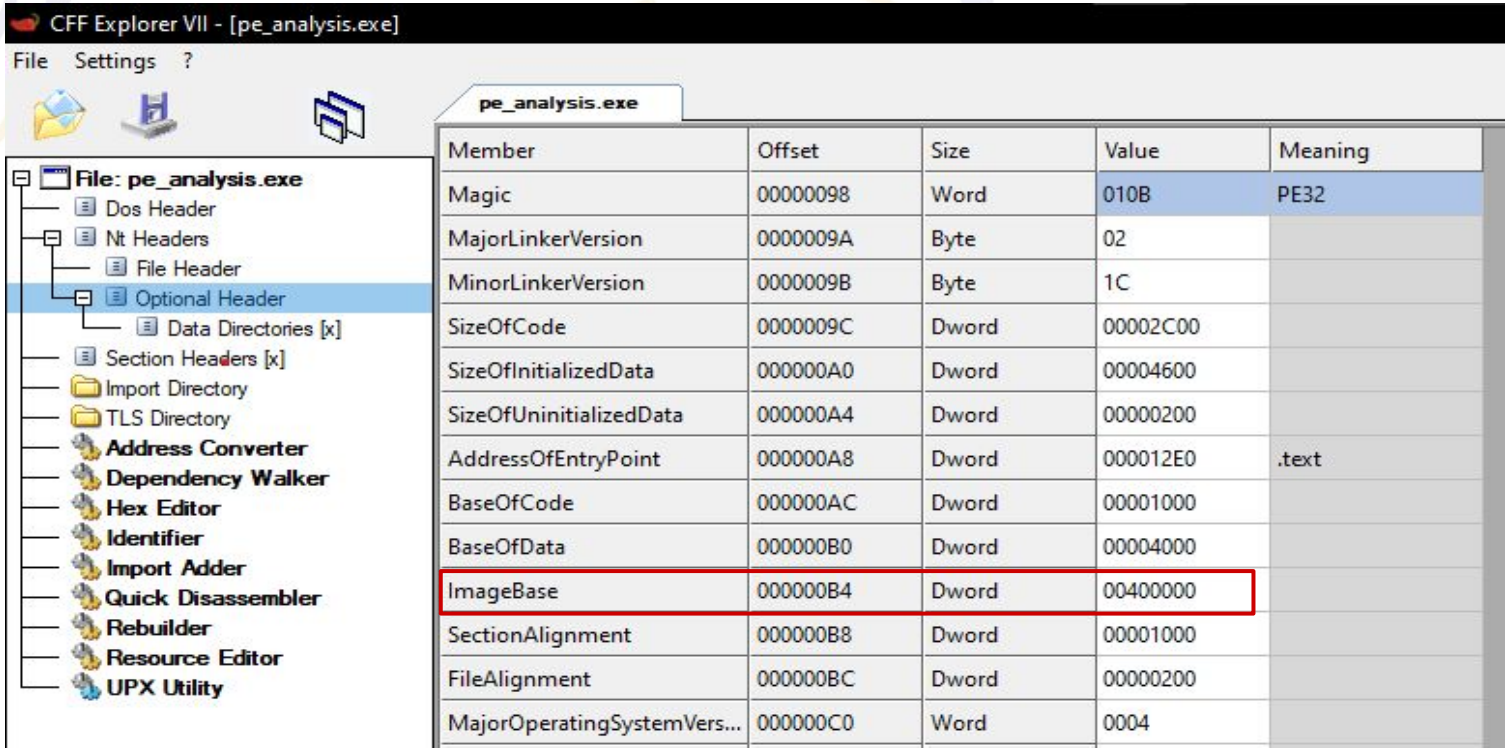



Practice Time !!


Challenge 1

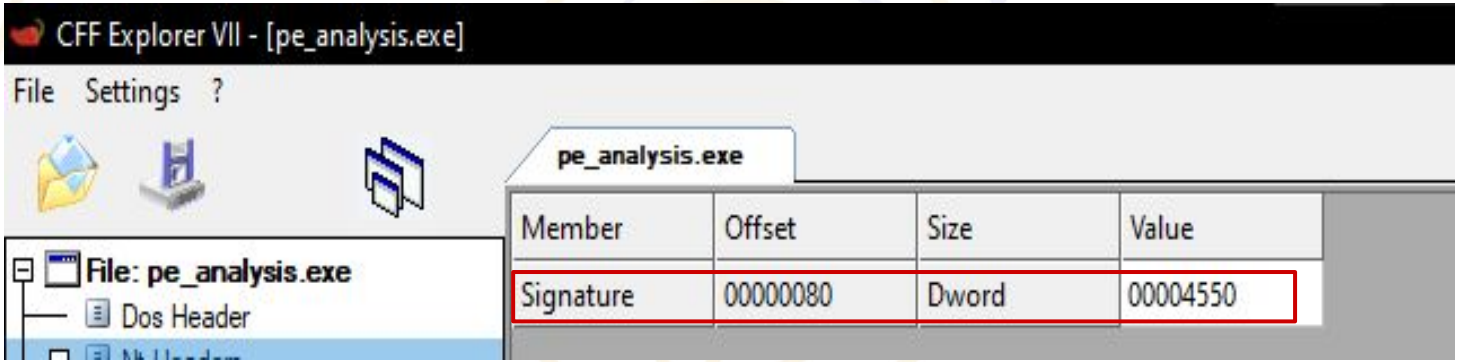

Time for Walkthrough!

What is the value of `ImageBase` of this executable?

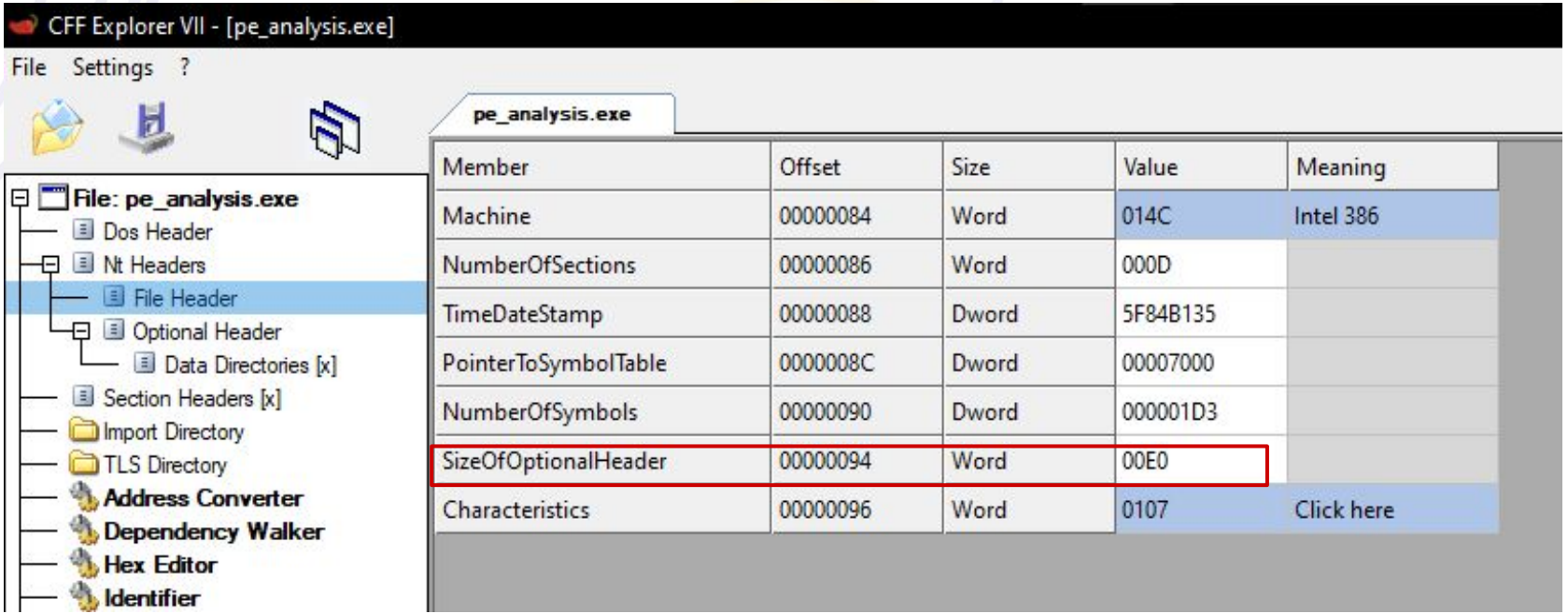


Member	Offset	Size	Value	Meaning
Magic	00000098	Word	010B	PE32
MajorLinkerVersion	0000009A	Byte	02	
MinorLinkerVersion	0000009B	Byte	1C	
SizeOfCode	0000009C	Dword	00002C00	
SizeOfInitializedData	000000A0	Dword	00004600	
SizeOfUninitializedData	000000A4	Dword	00000200	
AddressOfEntryPoint	000000A8	Dword	000012E0	.text
BaseOfCode	000000AC	Dword	00001000	
BaseOfData	000000B0	Dword	00004000	
ImageBase	000000B4	Dword	00400000	
SectionAlignment	000000B8	Dword	00001000	
FileAlignment	000000BC	Dword	00000200	
MajorOperatingSystemVers...	000000C0	Word	0004	

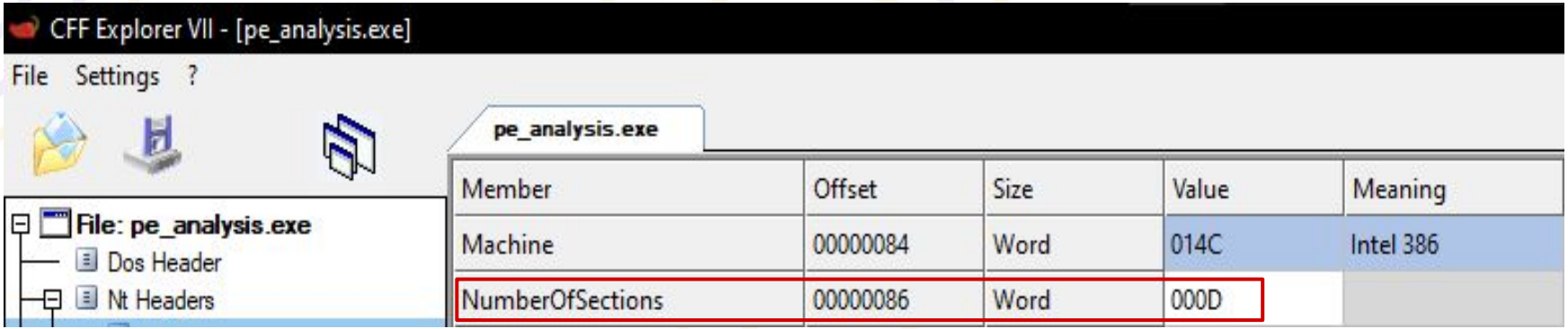
What is value in the `signature` field of PE header?



What is the value of `SizeOfOptionalHeader` field?



Number of sections:



Why is it necessary to know ?

- Fixing distorted headers.
- For unpacking the executable.
- Retrieving other important information related to the executable.
- Research purpose
  - For classifying an application as Malicious or Benign.

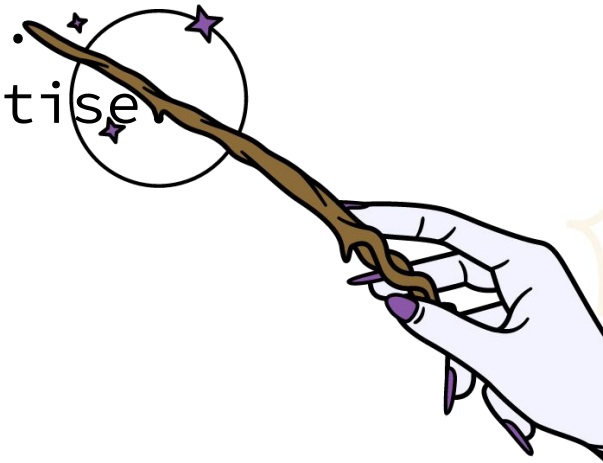
Why is it necessary to know ?

Index	Key Features	Malware (5598)	Normal (1237)	Difference
1	Size Of Initialized Data == 0	1626 (29%)	0 (0%)	29%
2	Unknown Section Name	2709 (48.4%)	16 (1.3%)	47.1%
3	DLL Characteristics == 0	5335 (95.3%)	401 (32.4%)	62.9%
4	Major Image Version == 0	5305 (94.8%)	486 (39.3%)	55.5%
5	Checksum == 0	5084 (90.8%)	474 (38.3%)	52.5%

http://cobweb.cs.uga.edu/~liao/PE_Final_Report.pdf

Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.



Contents

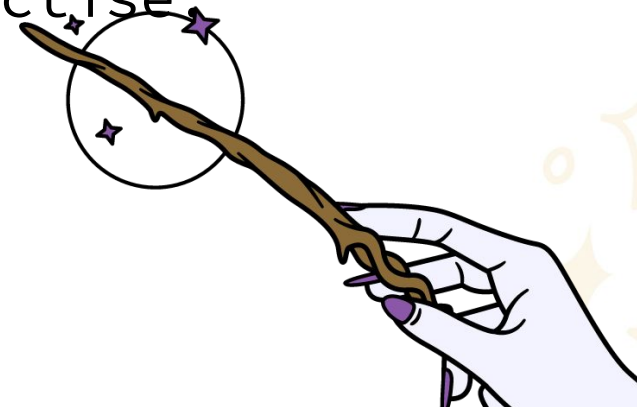
- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
  - Isolation of environment
  - Static & Dynamic Analysis
- ★ Windows Reversing and practise.

Steps for malware analysis

- Environment Isolation: Virtual Machines
- Basic Analysis
  - Fingerprint
  - Virus Scanning
  - Strings
  - Checking for packers and obfuscation
- Static Analysis
  - Analysing the disassembly and use tools for decompilation. Eg Ghidra, IDA pro
- Dynamic Analysis
  - Executing the sample
  - Analysis using debuggers

Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.



Contents

- ★ What is a Malware?
- ★ Why is reverse engineering necessary?
- ★ PE file structure.
- ★ Steps for Malware analysis.
- ★ Windows Reversing and practise.
  - x64dbg for reversing windows applications
  - Unpacking packed binaries
  - DLL Injection


Practice Time !!


Challenge 2

On executing

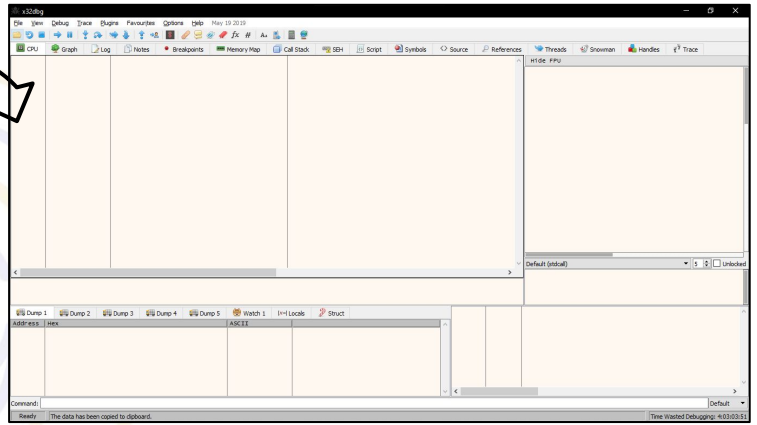
```
C:\Users\hp\Desktop\workshoptemp\challenge_file\Challenge-1>
C:\Users\hp\Desktop\workshoptemp\challenge_file\Challenge-1>challenge1.exe
Hello there!
Here is your flag:
Not so simple!!
```

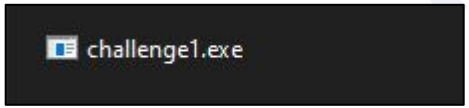

Time for Walkthrough!

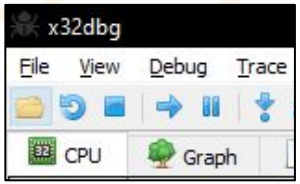

Challenge 2

Open x32dbg


x32dbg.exe
App




challenge1.exe


x32dbg
File
View
Debug
Trace
CPU
Graph

Open x32dbg

challenge1.exe - PID: 32A8 - Module: challenge1.exe - Thread: Main Thread 32D0 - x32dbg

File View Debug Trace Plugins Favourites Options Help May 19 2019

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Snowman Handles Trace

```
00401450    56          push esi
00401451    53          push ebx
00401452    83EC 14     sub esp,14
00401455    833D 28504000 cmp dword ptr ds:[405028],2
0040145C    8B4424 24   mov eax,dword ptr ss:[esp+24]
00401460    74 0A       je challenge1.40146C
00401462    C705 28504000 mov dword ptr ds:[405028],2
0040146C    83F8 02     cmp eax,2
0040146F    74 12       je challenge1.401483
00401471    83F8 01     cmp eax,1
00401474    74 42       je challenge1.4014B8
00401476    83C4 14     add esp,14
00401479    B8 01000000 mov eax,1
0040147E    5B          pop ebx
0040147F    5E          pop esi
00401480    C2 0C00     ret C
00401483    BE 14704000 mov esi,challenge1.407014
00401488    81EE 14704000 sub esi,challenge1.407014
0040148E    C1FE 02     sar esi,2
00401491    85F6        test esi,esi
00401493    7E E1       jle challenge1.401476
00401495    31DB        xor ebx,ebx
00401497    8B049D 14704000 mov eax,dword ptr ds:[ebx*4+407014]
0040149E    85C0        test eax,eax
004014A0    74 02       je challenge1.4014A4
004014A2    FFD0        call eax
004014A4    83C3 01     add ebx,1
004014A7    39F3        cmp ebx,esi
004014A9    75 EC       jne challenge1.401497
004014AB    83C4 14     add esp,14
004014AE    B8 01000000 mov eax,1
004014B3    5B          pop ebx
004014B4    5E          pop esi
004014B5    C2 0C00     ret C
004014B8    8B4424 28   mov eax,dword ptr ss:[esp+28]
```

```
EAX   00000000
EBX   00000000
ECX   00401450      challenge1.00401450
EDX   00400000      challenge1.00400000
EBP   0060F988
ESP   0060F99C
ESI   0060F9AC
EDI   00401450      challenge1.00401450

EIP   00401450      challenge1.00401450

EFLAGS   00000244
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 Empty 0.000000000000000
ST(1) 00000000000000000000 x87r1 Empty 0.000000000000000
ST(2) 00000000000000000000 x87r2 Empty 0.000000000000000
ST(3) 00000000000000000000 x87r3 Empty 0.000000000000000
ST(4) 00000000000000000000 x87r4 Empty 0.000000000000000
```

Hide FPU

```
esi=0060F9AC

.text:00401450 challenge1.exe:$1450 #850
```

Default (stdcall) 5 Unlocked

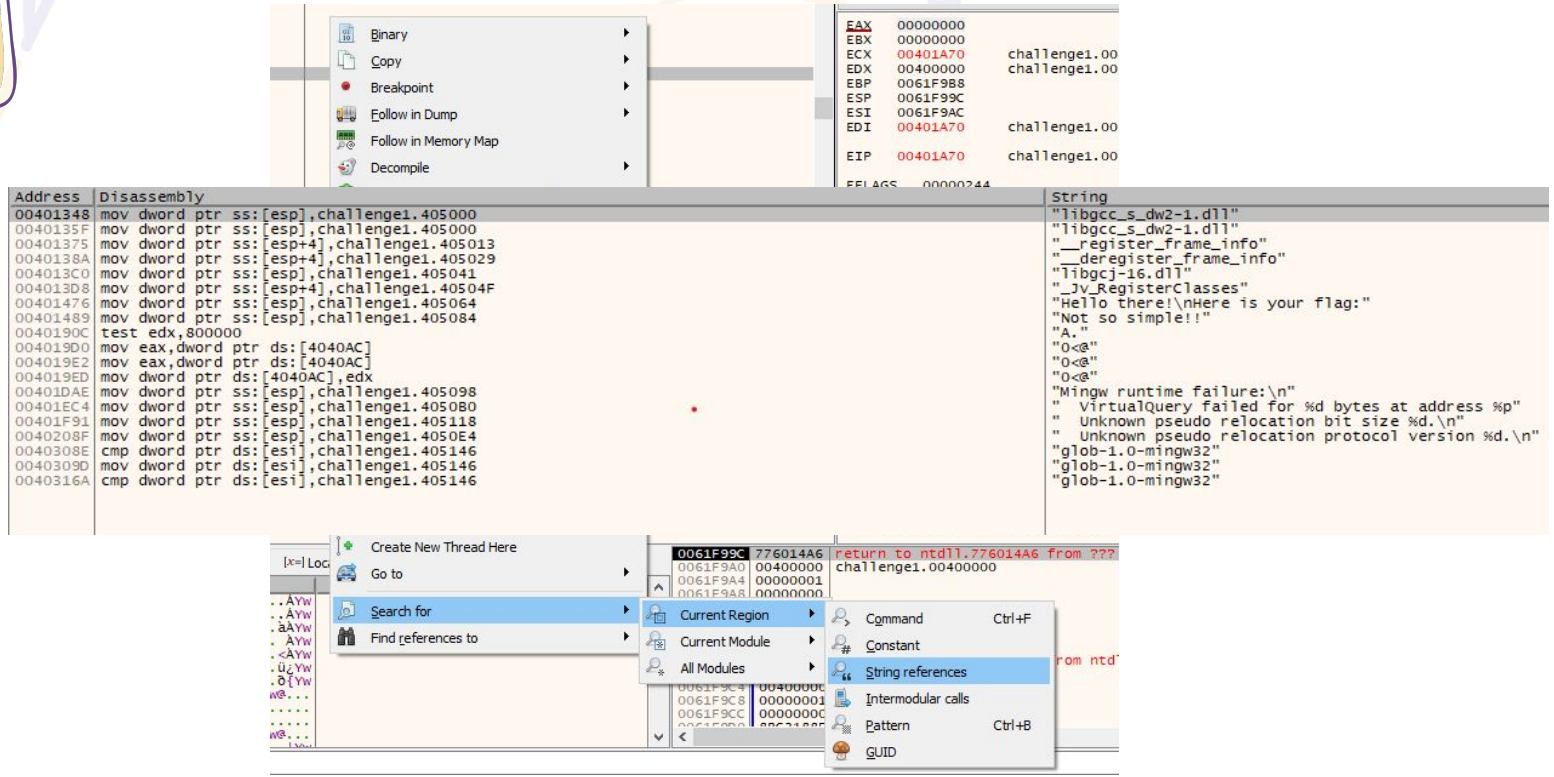
```
1: [esp+4] 00400000 challenge1.00400000
2: [esp+8] 00000001
3: [esp+C] 00000000
4: [esp+10] 00000000
5: [esp+14] 00401450 challenge1.00401450
```

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x=] Locals Struct

Address	Hex	ASCII
77591000	36 00 38 00 A0 C1 59 77 1C 00 1E 00 80 C1 59 77	6.8. ÁYw.....ÁYw
77591010	28 00 2A 00 54 C1 59 77 34 00 36 00 1C C1 59 77	(.*.TÁYw4.6..ÁYw
77591020	1E 00 20 00 FC C0 59 77 1A 00 1C 00 E0 C0 59 77	.. .üÀYw....àÀYw
77591030	18 00 1A 00 C4 C0 59 77 20 00 22 00 A0 C0 59 77	....ÄÀYw .". ÀYw
77591040	30 00 32 00 6C C0 59 77 2C 00 2E 00 3C C0 59 77	0.2.lÀYw,...<ÀYw
77591050	20 00 22 00 18 C0 59 77 18 00 1A 00 FC BF 59 77	.".. ÀYw....ü¿Yw
77591060	10 00 12 00 E8 BF 59 77 24 00 26 00 F0 7B 59 77	....è¿Yw$.&.ð{Yw
77591070	18 00 00 00 00 00 00 00 84 17 59 77 40 00 00 00	..........Yw@...
77591080	00 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00	....
77591090	7C 17 59 77 40 00 00 00 00 00 00 00 00 00 00 00	|.Yw@...........
775910A0	18 00 00 00 00 00 00 00 94 17 59 77 40 00 00 00	..........Yw@...

```
0060F99C 776014A6 return to ntdll.776014A6 from ???
0060F9A0 00400000 challenge1.00400000
0060F9A4 00000001
0060F9A8 00000000
0060F9AC 00000000
0060F9B0 00401450 challenge1.00401450
0060F9B4 00400000 challenge1.00400000
0060F9B8 0060FA04
0060F9BC 775CFEE7 return to ntdll.775CFEE7 from ntdll.77601490
0060F9C0 00401450 challenge1.00401450
0060F9C4 00400000 challenge1.00400000
0060F9C8 00000001
0060F9CC 00000000
```

Open x32dbg: check for strings



Open x32dbg: check the function

CPU | Graph | Log | Notes | Breakpoints | Memory Map | Call Stack | SEH | Script | Symbols | Source

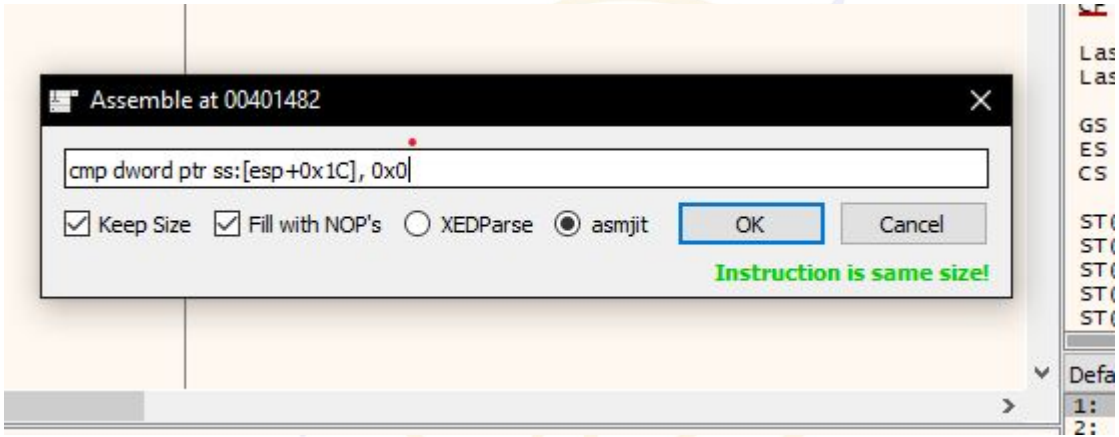
```
0040145C    C9              leave
0040145D    C3              ret
0040145E    90              nop
0040145F    90              nop
00401460    55              push ebp
00401461    89E5            mov ebp,esp
00401463    83E4 F0         and esp,FFFFFFF0
00401466    83EC 20         sub esp,20
00401469    E8 E2050000     call challenge1.401A50
0040146E    C74424 1C 00(   mov dword ptr ss:[esp+1C],0
00401476    C70424 645040(  mov dword ptr ss:[esp],challenge1.405064      405064:"Hello there!\nHere is your flag:"
0040147D    E8 6E260000     call <JMP.&puts>
00401482    837C24 1C 00    cmp dword ptr ss:[esp+1C],0
00401487    75 0E           jne challenge1.401497
00401489    C70424 845040(  mov dword ptr ss:[esp],challenge1.405084      405084:"Not so simple!!"
00401490    E8 5B260000     call <JMP.&puts>
00401495    EB 05           jmp challenge1.40149C
00401497    E8 07000000     call challenge1.4014A3
0040149C    B8 00000000     mov eax,0
004014A1    C9              leave
004014A2    C3              ret
004014A3    55              push ebp
004014A4    89E5            mov ebp,esp
004014A6    57              push edi
004014A7    56              push esi
004014A8    53              push ebx
004014A9    81EC AC000000(  sub esp,AC
004014AF    8D85 7CFFFFFF(  lea eax,dword ptr ss:[ebp-84]
004014B5    BB 20404000     mov ebx,challenge1.404020
004014BA    BA 1A000000     mov edx,1A
004014BF    89C7            mov edi,eax
004014C1    89DE            mov esi,ebx
004014C3    89D1            mov ecx,edx
004014C5    F3:A5           rep movsd
004014C7    C745 E4 0000(   mov dword ptr ss:[ebp-1C],0
004014CE    EB 20           jmp challenge1.4014F0
```

What to Patch ?

Figure it out

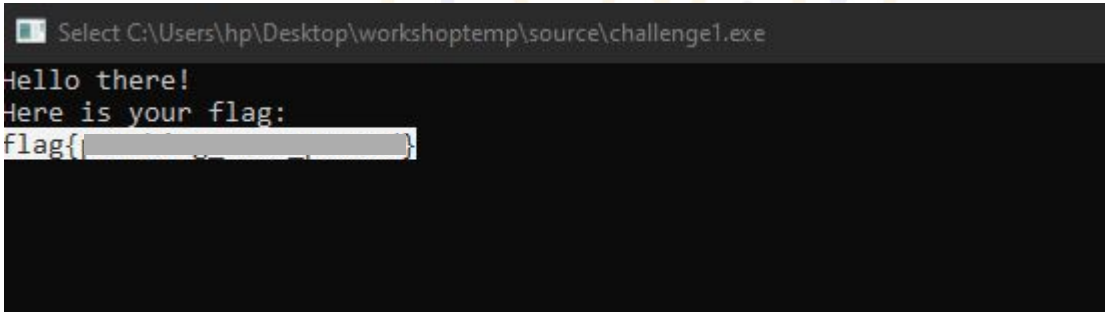
```
00401476    C70424 645040  mov dword ptr ss:[esp],challenge1.405064
0040147D    E8 6E260000    call <JMP.&puts>
00401482    837C24 1C 00   cmp dword ptr ss:[esp+1C],0
00401487    75 0E          jne challenge1.401497
00401489    C70424 845040  mov dword ptr ss:[esp],challenge1.405084
```

Figure it out




```
0040146E    C74424 1C 00(  mov dword ptr ss:[esp+1C],0
00401476    C70424 645040( mov dword ptr ss:[esp],challenge1.405064
0040147D    E8 6E260000    call <JMP.&puts>
00401482    837C24 1C 01   cmp dword ptr ss:[esp+1C],1
00401487    75 0E          jne challenge1.401497
00401489    C70424 845040( mov dword ptr ss:[esp],challenge1.405084
00401490    E8 5B260000    call <JMP.&puts>
00401495    EB 05          jmp challenge1.40149C
```

On executing



Packing

- ## What is Packer ?

  A packer is a software that will compress your executable files, similar to the way zip files work.
  When you run the executable it will run the uncompressed code which unpacks the rest of the code and runs it.

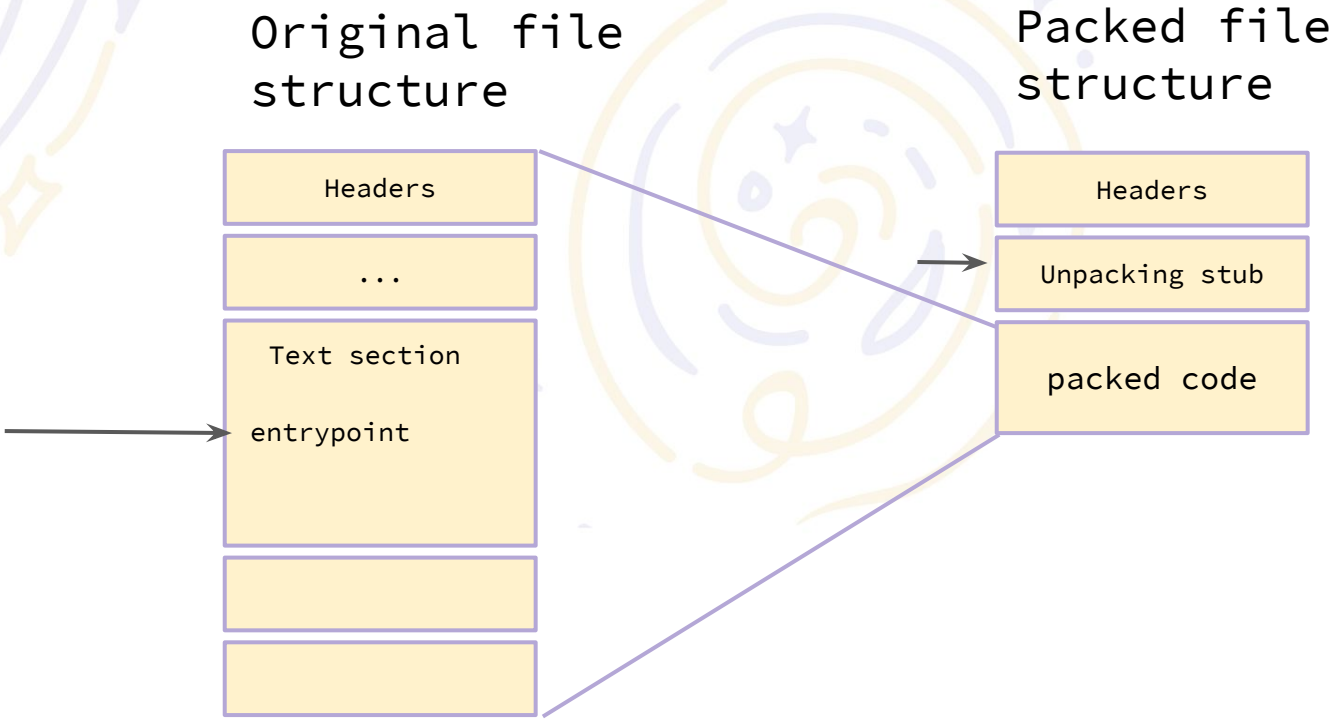
- ## Why to pack any application?
  - To decrease the size of the file.
  - To obfuscate it.
  - Difficult to reverse engineer.

How is Packing done?

The the original code goes through the packing process, where it is compressed or encrypted.

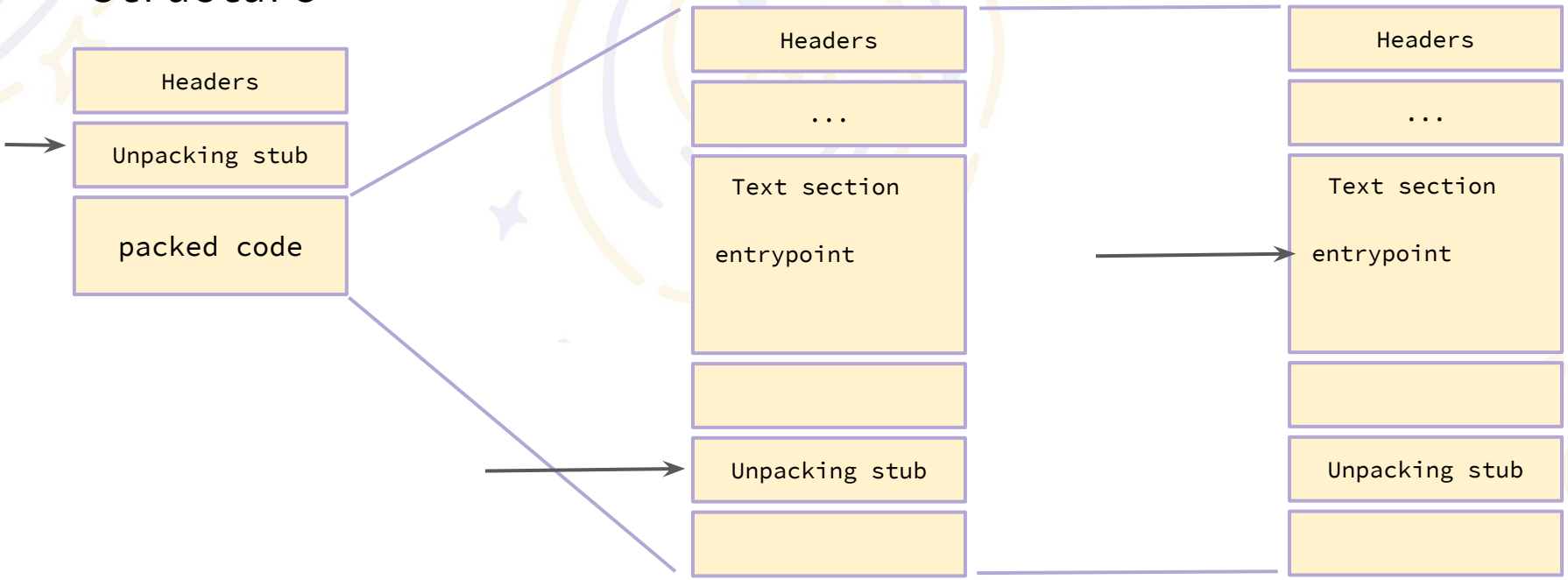
- This new executable contains:
  - New headers
  - Decompression stub
  - Packed data
- The original entry point is changed.

How is Packing done?



How is Unpacking done?

Packed file structure



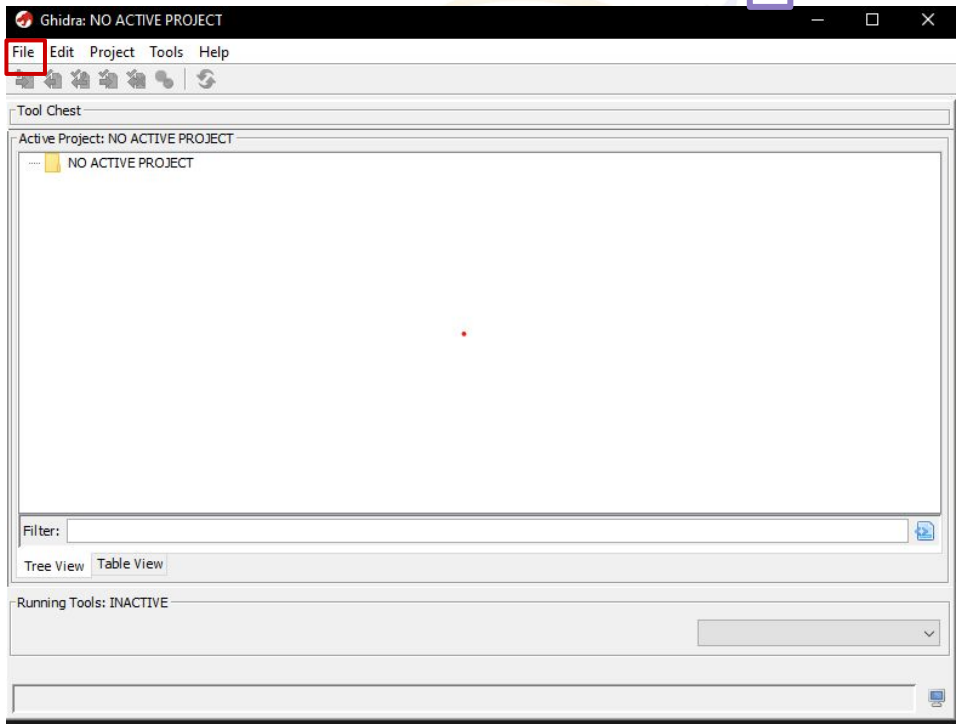

Challenge 3

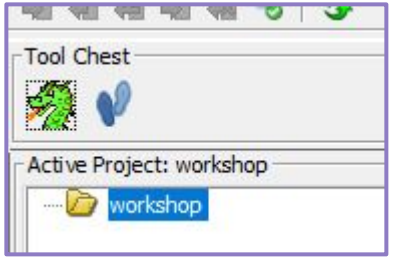
On executing

```
C:\Users\hp\Desktop\workshoptemp\challenge_file\Challenge-2>challenge2.exe
Hello there!
Enter the password:
```


Challenge 3

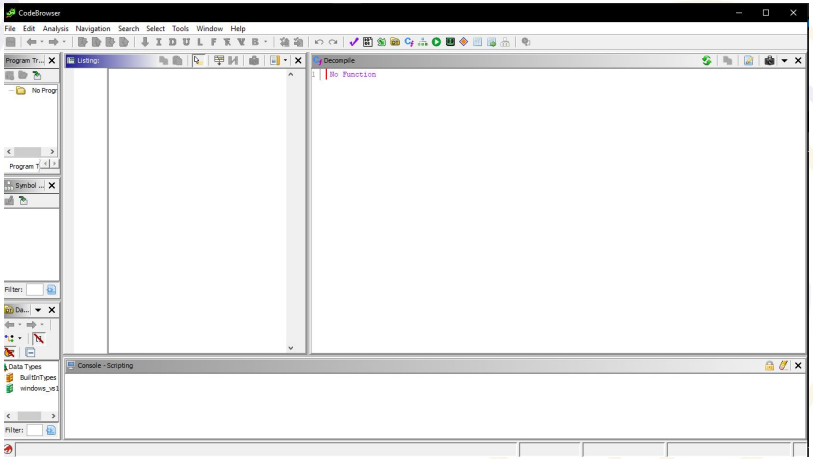
Open Ghidra

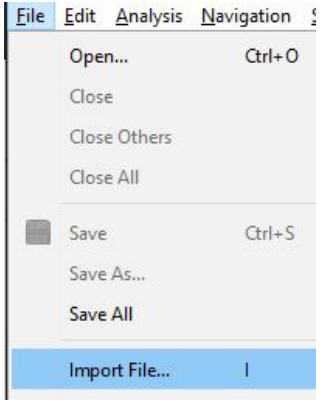

Ghidra: NO ACTIVE PROJECT
File
Edit
Project
Tools
Help
Tool Chest
Active Project: NO ACTIVE PROJECT
NO ACTIVE PROJECT
Filter:
Tree View
Table View
Running Tools: INACTIVE

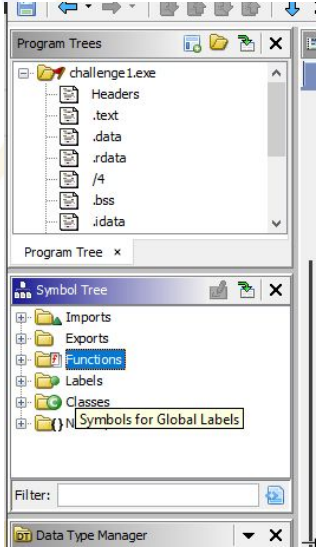

Tool Chest
Active Project: workshop
workshop


Challenge 3

Open Ghidra


CodeBrowser
File Edit Analysis Navigation Search Select Tools Window Help
Listing:
Decompile
No Function
Console - Scripting

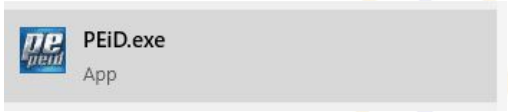

File Edit Analysis Navigation
Open... Ctrl+O
Close
Close Others
Close All
Save Ctrl+S
Save As...
Save All
Import File...

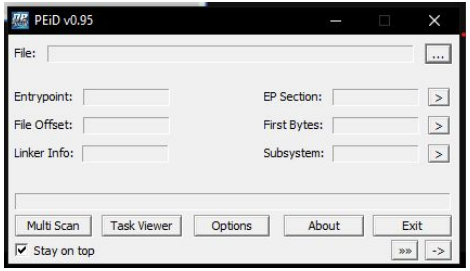

Program Trees
challenge1.exe
Headers
.text
.data
.rdata
/4
.bss
.idata
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Symbols for Global Labels
Filter:
Data Type Manager

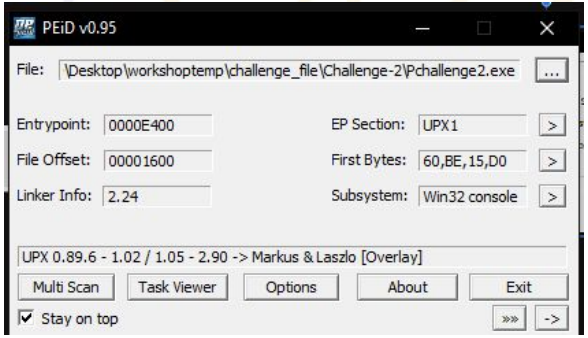

Symbol Tree
Imports
Exports
Functions
entry
Labels
Classes
Namespaces
Filter:


Challenge 3

PEiD


PEiD.exe
App


PEiD v0.95
File:
Entrypoint:
EP Section:
File Offset:
First Bytes:
Linker Info:
Subsystem:
Multi Scan
Task Viewer
Options
About
Exit
Stay on top


PEiD v0.95
File: \Desktop\workshoptemp\challenge_file\Challenge-2\Pchallenge2.exe
Entrypoint: 0000E400
EP Section: UPX1
File Offset: 00001600
First Bytes: 60,BE,15,D0
Linker Info: 2.24
Subsystem: Win32 console
UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo [Overlay]
Multi Scan
Task Viewer
Options
About
Exit
Stay on top

Demo

Pchallenge2.exe - PID: 3964 - Module: pchallenge2.exe - Thread: Main Thread 232C - x32dbg

File View Debug Trace Plugins Favourites Options Help May 19 2019

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References

```
0040E564    57              push edi                                edi:EntryPoint
0040E565    FFD5            call ebp
0040E567    8D87 9F010000   lea eax,dword ptr ds:[edi+19F]
0040E56D    8020 7F         and byte ptr ds:[eax],7F
0040E570    8060 28 7F      and byte ptr ds:[eax+28],7F
0040E574    58              pop eax
0040E575    50              push eax
0040E576    54              push esp
0040E577    50              push eax
0040E578    53              push ebx
0040E579    57              push edi                                edi:EntryPoint
0040E57A    FFD5            call ebp
0040E57C    58              pop eax
0040E57D    8D9E 00F0FFFF   lea ebx,dword ptr ds:[esi-1000]
0040E583    8DBB A9E50000   lea edi,dword ptr ds:[ebx+E5A9]         edi:EntryPoint
0040E589    57              push edi                                edi:EntryPoint
0040E58A    31C0            xor eax,eax
0040E58C    AA              stosb
0040E58D    59              pop ecx                                 ecx:EntryPoint
0040E58E    49              dec ecx                                 ecx:EntryPoint
0040E58F    50              push eax
0040E590    6A 01           push 1
0040E592    53              push ebx
0040E593    FFD1            call ecx                                ecx:EntryPoint
0040E595    61              popad
0040E596    8D4424 80       lea eax,dword ptr ss:[esp-80]
0040E59A    6A 00           push 0
0040E59C    39C4            cmp esp,eax
0040E59E    75 FA           jne pchallenge2.40E59A
0040E5A0    83EC 80         sub esp,FFFFFF80
0040E5A3    E9 D82CFFFF     jmp pchallenge2.401280
0040E5A8    EB 00           jmp pchallenge2.40E5AA
0040E5AA    56              push esi                                esi:EntryPoint
0040E5AB    BE 04704000     mov esi,pchallenge2.407004              esi:EntryPoint
0040E5B0    FC              cld
0040E5B1    AD              lodsd
```

Demo

Pchallenge2.exe - PID: 3964 - Module: pchallenge2.exe - Thread: Main Thread 232C - x32dbg

File View Debug Trace Plugins Favourites Options Help May 19 2019

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Snowman Handles Trace

```
00401350    55          push ebp
00401351    89E5        mov ebp,esp
00401353    83E4 F0     and esp,FFFFFFF0
00401356    83EC 30     sub esp,30
00401359    E8 62060000 call pchallenge2.4019C0
0040135E    C74424 1E 73 mov dword ptr ss:[esp+1E],6E663773
00401366    66:C74424 22 mov word ptr ss:[esp+22],6736
0040136D    C64424 24 00 mov byte ptr ss:[esp+24],0
00401372    C64424 24 00 mov byte ptr ss:[esp+24],0
00401377    C70424 243040 mov dword ptr ss:[esp],pchallenge2.403024   403024:"Hello there!\nEnter the password:"
0040137E    E8 AD080000 call <JMP.&puts>
00401383    8D4424 25   lea eax,dword ptr ss:[esp+25]
00401387    894424 04   mov dword ptr ss:[esp+4],eax
0040138B    C70424 453040 mov dword ptr ss:[esp],pchallenge2.403045   403045:"%s"
00401392    E8 A1080000 call <JMP.&scanf>
00401397    C64424 2B 00 mov byte ptr ss:[esp+2B],0
0040139C    C74424 2C 000 mov dword ptr ss:[esp+2C],0
004013A4    EB 44       jmp pchallenge2.4013EA
004013A6    8D5424 25   lea edx,dword ptr ss:[esp+25]
004013AA    8B4424 2C   mov eax,dword ptr ss:[esp+2C]
004013AE    01D0        add eax,edx
004013B0    0FB600      movzx eax,byte ptr ds:[eax]
004013B3    0FBEC0      movsx eax,al
004013B6    8D48 03     lea ecx,dword ptr ds:[eax+3]
004013B9    8D5424 1E   lea edx,dword ptr ss:[esp+1E]
004013BD    8B4424 2C   mov eax,dword ptr ss:[esp+2C]
004013C1    01D0        add eax,edx
004013C3    0FB600      movzx eax,byte ptr ds:[eax]
004013C6    0FBEC0      movsx eax,al
004013C9    39C1        cmp ecx,eax
004013CB    74 18       je pchallenge2.4013E5
004013CD    C70424 483040 mov dword ptr ss:[esp],pchallenge2.403048   403048:"Wrongg!!"
004013D4    E8 57080000 call <JMP.&puts>
004013D9    C70424 000000 mov dword ptr ss:[esp],0
004013E0    E8 5B080000 call <JMP.&exit>
```

```
EAX   00000001
EBX   002D5000
ECX   00401940      pchallenge2.00401940
EDX   00000080
EBP   0060FF80
ESP   0060FF0C
ESI   0040E400      <pchallenge2.EntryPoint>
EDI   0040E400      <pchallenge2.EntryPoint>

EIP   00401350      pchallenge2.00401350

EFLAGS   00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError   00000000 (ERROR_SUCCESS)
LastStatus C00700BB

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 Empty 0.000000000000000
ST(1) 00000000000000000000 x87r1 Empty 0.000000000000000
ST(2) 00000000000000000000 x87r2 Empty 0.000000000000000
ST(3) 00000000000000000000 x87r3 Empty 0.000000000000000
ST(4) 00000000000000000000 x87r4 Empty 0.000000000000000
```

Hide FPU

Default (stdcall) 5 Unlocked

```
1: [esp+4]  00000001
2: [esp+8]  00BF2260 &"C:\\Users\\hp\\Desktop\\workshoptem
3: [esp+C]  00BF17A8 &"ALLUSERSPROFILE=C:\\ProgramData"
4: [esp+10]  00405000 pchallenge2.00405000
5: [esp+14]  0060FF40
```

```
ebp=0060FF80

UPXO:00401350 pchallenge2.exe:$1350 #0
```

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x=] Locals Struct

```
Address   Hex                                              ASCII
77591000  36 00 38 00 A0 C1 59 77 1C 00 1E 00 80 C1 59 77  6.8. ÁYw.....ÁYw
77591010  28 00 2A 00 54 C1 59 77 34 00 36 00 1C C1 59 77  (.*.TÁYw4.6..ÁYw
77591020  1E 00 20 00 FC C0 59 77 1A 00 1C 00 E0 C0 59 77  .. .üÀYw....àÀYw
77591030  18 00 1A 00 C4 C0 59 77 20 00 22 00 A0 C0 59 77  ....ÄÀYw .". ÀYw
77591040  30 00 32 00 6C C0 59 77 2C 00 2E 00 3C C0 59 77  0.2.lÀYw,...<ÀYw
77591050  20 00 22 00 18 C0 59 77 18 00 1A 00 FC BF 59 77   .".ÀYw....ü¿Yw
77591060  10 00 12 00 E8 BF 59 77 24 00 26 00 E0 78 59 77  ....è¿Yw$.&.àxYw
77591070  18 00 00 00 00 00 00 00 84 17 59 77 40 00 00 00  ..........Yw@...
77591080  00 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00  ................
77591090  7C 17 59 77 40 00 00 00 00 00 00 00 00 00 00 00  |.Yw@...........
775910A0  18 00 00 00 00 00 00 00 94 17 59 77 40 00 00 00  ..........Yw@...
```

```
0060FF0C  004010FD return to pchallenge2.004010FD from pchallenge2.00401350
0060FF10  00000001
0060FF14  00BF2260 &"C:\\Users\\hp\\Desktop\\workshoptemp\\challenge_file\\c
0060FF18  00BF17A8 &"ALLUSERSPROFILE=C:\\ProgramData"
0060FF1C  00405000 pchallenge2.00405000
0060FF20  0060FF40
0060FF24  FFFFFFFF
0060FF28  0060FF44
0060FF2C  759CCCC0 msvcrt.759CCCC0
0060FF30  72B97107
0060FF34  FFFFFFFE
0060FF38  0060FF44
0060FF3C  759CCA67 return to msvcrt.759CCA67 from msvcrt.759CC9DC
```

Command: Default

Paused INT3 breakpoint at pchallenge2.004010F8 (004010F8)! Time Wasted Debugging: 4:05:19:43

Demo

```
00401387    894424 04       mov dword ptr ss:[esp+4],eax
0040138B    C70424 453040   mov dword ptr ss:[esp],pchallenge2.403045
00401392    E8 A1080000     call <JMP.&scanf>
00401397    C64424 2B 00    mov byte ptr ss:[esp+2B],0
0040139C    C74424 2C 000   mov dword ptr ss:[esp+2C],0
004013A4    EB 44           jmp pchallenge2.4013EA
004013A6    8D5424 25       lea edx,dword ptr ss:[esp+25]
004013AA    8B4424 2C       mov eax,dword ptr ss:[esp+2C]
004013AE    01D0            add eax,edx
004013B0    0FB600          movzx eax,byte ptr ds:[eax]
004013B3    0FBEC0          movsx eax,al
004013B6    8D48 03         lea ecx,dword ptr ds:[eax+3]
004013B9    8D5424 1E       lea edx,dword ptr ss:[esp+1E]
004013BD    8B4424 2C       mov eax,dword ptr ss:[esp+2C]
004013C1    01D0            add eax,edx
004013C3    0FB600          movzx eax,byte ptr ds:[eax]
004013C6    0FBEC0          movsx eax,al
004013C9    39C1            cmp ecx,eax
004013CB    74 18           je pchallenge2.4013E5
```

Some statistics : Packed Malwares

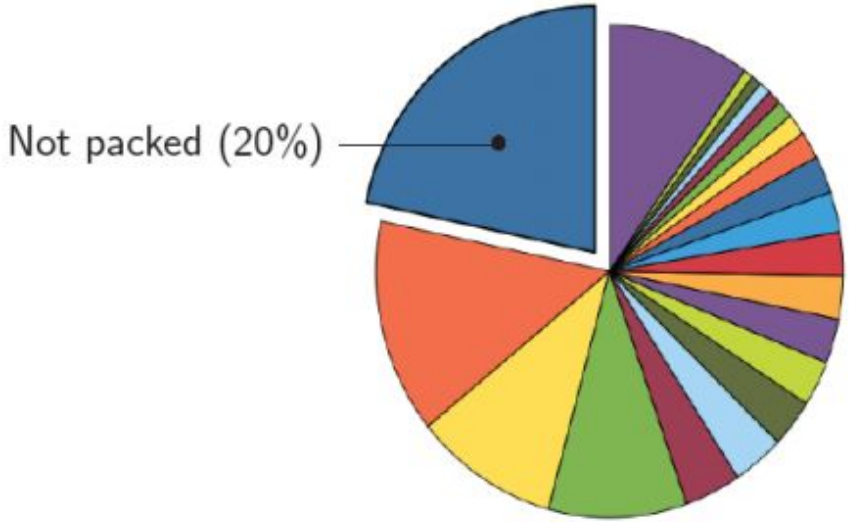


Fig. 1. Malware and packing, 80% of new malware are packed with various packers, 50% of new malware samples are simply repacked versions of existing malware [8].

https://www.researchgate.net/publication/307843262_Generic_Packing_Detection_Using_Several_Complexity_Analysis_for_Accurate_Malware_Detection

Demo

```
C:\Users\hp\Desktop\workshoptemp\challenge_file\Challenge-2>challenge2.exe
Hello there!
Enter the password:

correct!!!
```


Practice Time !!


Time for Walkthrough!

DLL

★ What is a DLL?

- ❏ Dynamic link library
- ❏ It is a library that contains code and data which can be used by more than one program at the same time.
- ❏ Shared library concept.
- ❏ Same file format as PE.
- ❏ Cannot execute on their own.

DLL

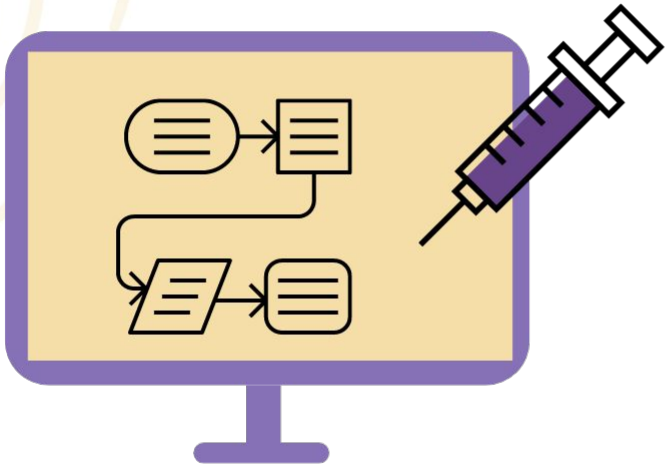
★ Advantages

- ❏ Modularisation of code.
- ❏ Low resource consumption.
- ❏ Eases the process of development/modification

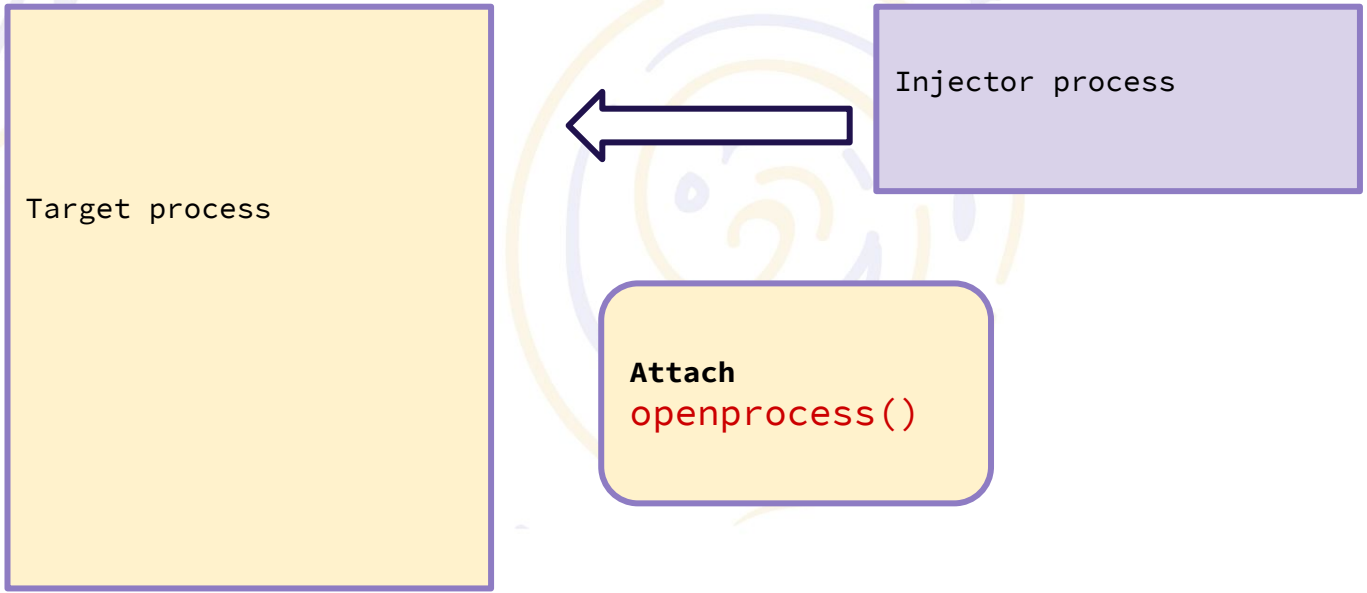
DLL Injection

★ DLL injection is a technique used for executing code within the program, by forcing it to load and run a dynamic library that was not actually meant by its original design.

- ❏ Injector process
- ❏ Target process

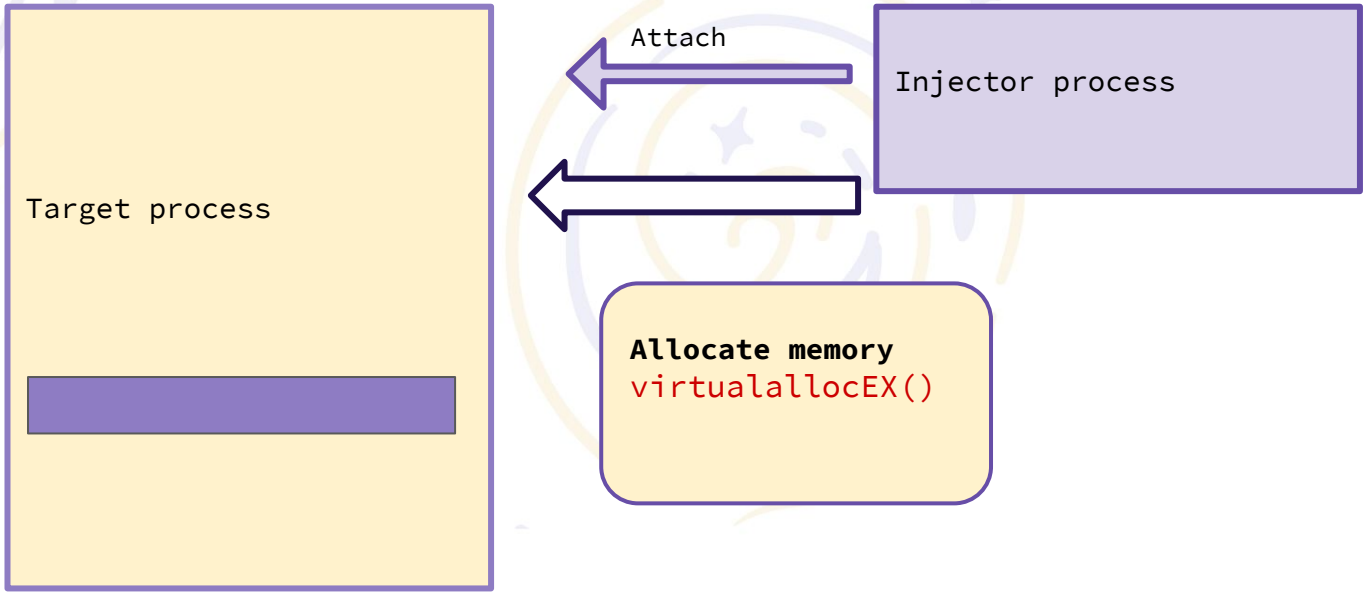


DLL Injection



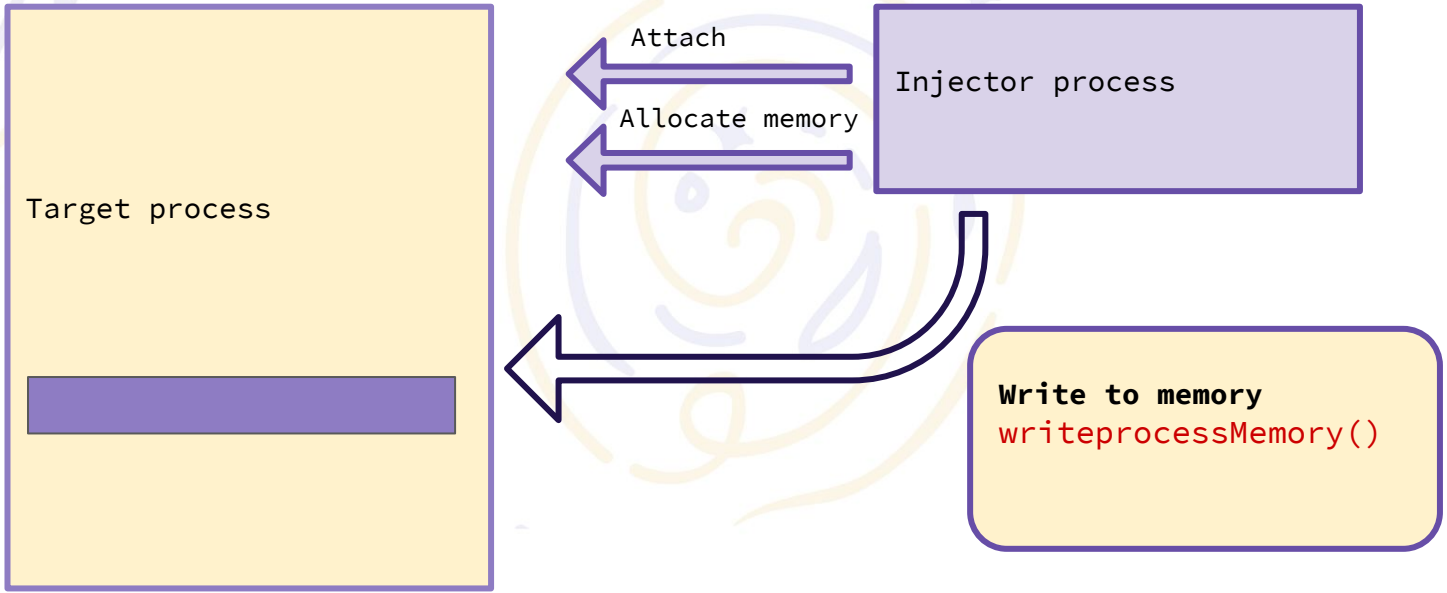

```
HANDLE process = OpenProcess(PROCESS_ALL_ACCESS, 0, process_id);
```

DLL Injection



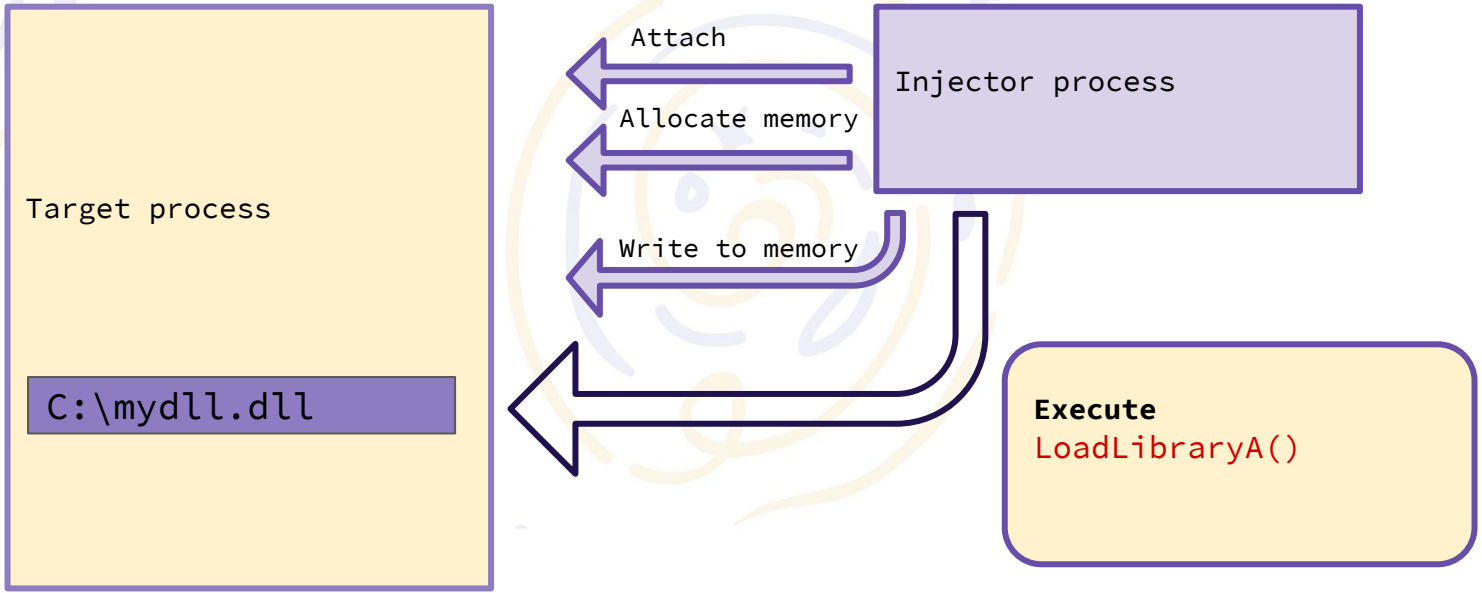

```
base_address = VirtualAllocEx(process, NULL, sizeof(pathDLL), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
```

DLL Injection



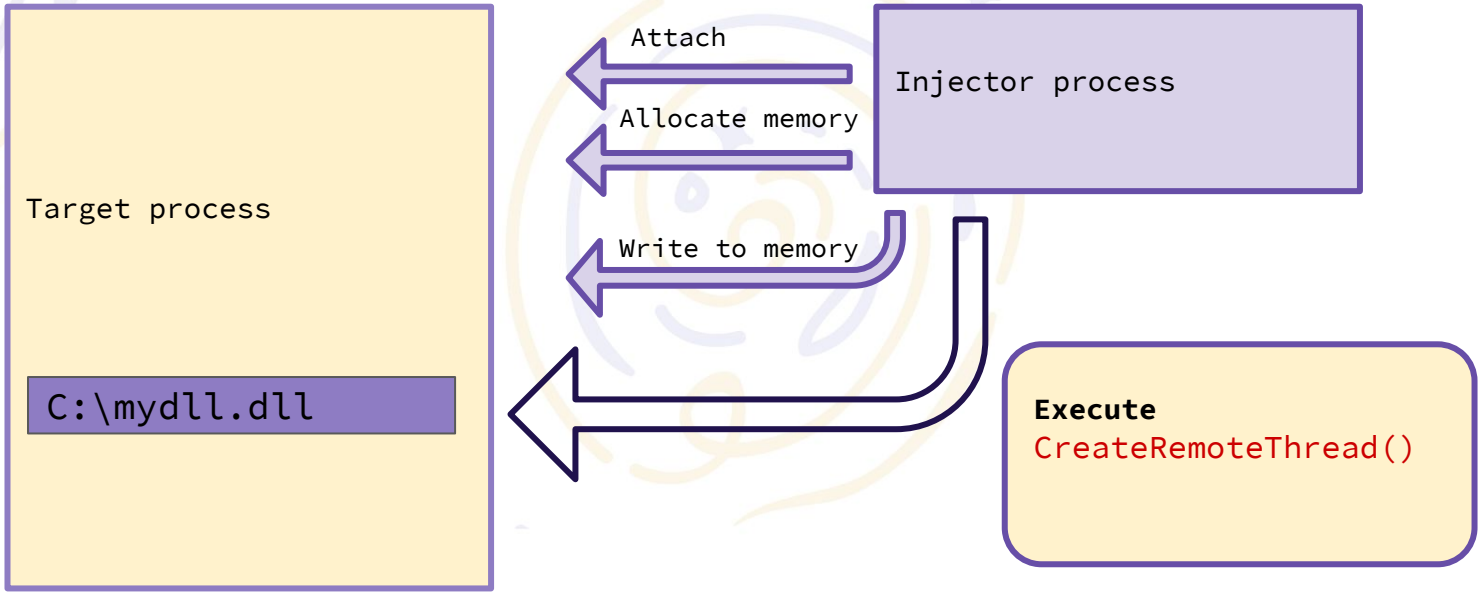

```
WriteProcessMemory(process, base_address, pathDLL, strlen(pathDLL), NULL)
```

DLL Injection




```
LPVOID addr = (LPVOID)GetProcAddress(GetModuleHandle("kernel32.dll"), "LoadLibraryA");
```

DLL Injection




```
HANDLE threadID = CreateRemoteThread(process, NULL,0,(LPTHREAD_START_ROUTINE)addr,base_address, 0, NULL);
```

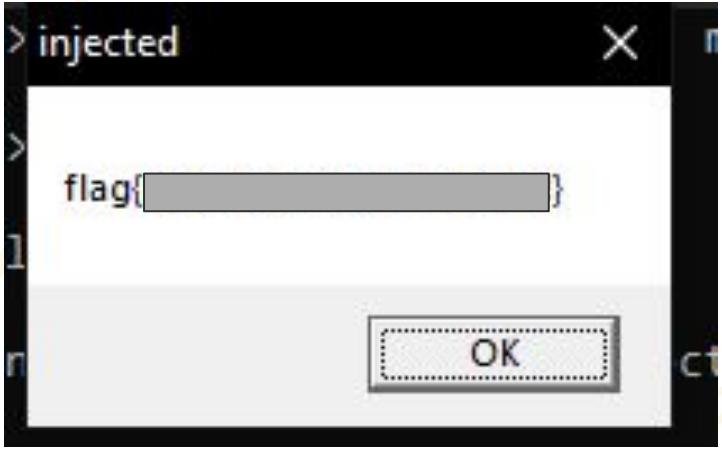
Tada!!


Challenge 4

Steps:

Inject.exe
Victim.exe
Mydll.dll

- ❏ Run Victim.exe
- ❏ Open procexp.exe
- ❏ Take the PID of victim.exe
- ❏ Use inject.exe to inject mydll.dll




Practice Time !!


Walkthrough
Time !!

Different Injection methods:

- DLL Injection
  - Reflective Dll injection
- PE Injection
- Process Hollowing

Points

- Malware use these concepts to attack.
- DLL inject can done on anti-virus softwares.
- Embed it into existing processes so that they can remain undetected and launch and execute additional successful attacks.
- Not all the Malwares are the same so be prepared for a new one.

Anything can happen at any moment

Points

- Danger is not just for a Personal Computer
- Weaponized malwares can affect:
  - Personal systems
  - Public infrastructures
  - Power grids
  - Nuclear plants
  - ...


Challenge 4


Bonus challenge!!

Challenge part 1

Find the 1st function

- TLScalback()
- Windows APIs:
  - Findresource()---> “BIN”
  - Loadresource()
  - SizeofResource()
  - Lockresource()
  - VirtualAlloc()
  - …
- Find the usage of the returned address

Challenge part 2

- Get the dropped DLL.
- Load it in ghidra.
- Get the IMP encryption algorithms

```
CryptAcquireContext dwProvType
CryptCreateHash Algid
CryptDeriveKey Algid
```

Challenge part 2

- Get the dropped DLL.
- Load it in ghidra.
- Get the IMP encryption algorithms

```
CryptAcquireContext dwProvType 0x18 PROV_RSA_AES
CryptCreateHash Algid 0x800C CALG_SHA_256
CryptDeriveKey Algid 0x660E CALG_AES_128
```

Challenge part 3

What to do????

Challenge part 3

Scripting??

Challenge part 3

Easy-peasy

- Swap the address

Challenge part 3

- Get the flag :P


Any
Questions
?

Thank you

&

Stay Safe