09/16/2025

Malicious Code Analysis

Fangtian Zhong
CSCI 591

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

Overview

01 Export

02 Relocation

03 Retrofitting

Part One

01

Export

2025-8-26

Data Directories

```
#define IMAGE_DIRECTORY_ENTRY_EXPORT          0   // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT          1   // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE        2   // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION       3   // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY        4   // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC       5   // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG           6   // Debug Directory
//      IMAGE_DIRECTORY_ENTRY_COPYRIGHT       7   // (X86 usage)
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE    7   // Architecture Specific Data
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR       8   // RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS             9   // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG     10  // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT    11  // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT             12  // Import Address Table
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT    13  // Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR  14  // COM Runtime descriptor
```

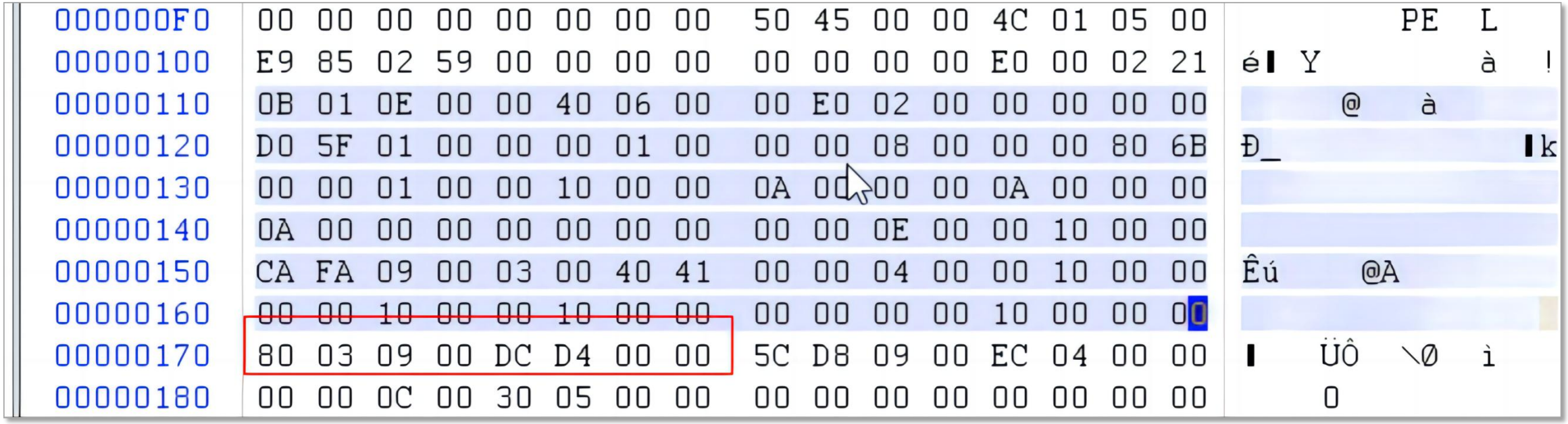
Data Directories

```
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   VirtualAddress;
    DWORD   Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

Export Directory Table

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;            // Reserved, must be 0.
    DWORD   TimeDateStamp;              //The time and date that the export data was created.
    WORD    MajorVersion;               //The major version number.
    WORD    MinorVersion;               //The minor version number.
    DWORD   Name;                       //The address of the ASCII string that contains the name of the DLL.
    DWORD   Base;                       //The starting ordinal number for exports in this image.
                                        //It is usually set to 1.
    DWORD   NumberOfFunctions;          //The number of entries in the export address table.
    DWORD   NumberOfNames;              //The number of entries in the name pointer table.
    DWORD   AddressOfFunctions;         //The address of the export address table.
    DWORD   AddressOfNames;             //The address of the export name pointer table.
    DWORD   AddressOfNameOrdinals;      //The address of the ordinal table, relative to the image base.
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

Export Directory Table



000000F0	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	05	00
00000100	E9	85	02	59	00	00	00	00	00	00	00	00	E0	00	02	21
00000110	0B	01	0E	00	00	40	06	00	00	E0	02	00	00	00	00	00
00000120	D0	5F	01	00	00	00	01	00	00	00	08	00	00	00	80	6B
00000130	00	00	01	00	00	10	00	00	0A	00	00	00	0A	00	00	00
00000140	0A	00	00	00	00	00	00	00	00	00	0E	00	00	10	00	00
00000150	CA	FA	09	00	03	00	40	41	00	00	04	00	00	10	00	00
00000160	00	00	10	00	00	10	00	00	00	00	00	00	10	00	00	00
00000170	80	03	09	00	DC	D4	00	00	5C	D8	09	00	EC	04	00	00
00000180	00	00	0C	00	30	05	00	00	00	00	00	00	00	00	00	00

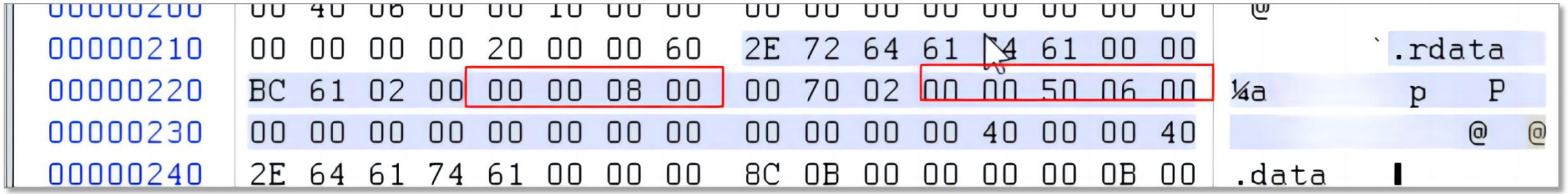
Virtual Address== 0x90380 Size == D4DC

Section Headers

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

Export Directory Table

- Because we don’t run the program, we should convert the Virtual Address (RVA) to File Address (FOA).



- In this case, 00080000 (Section VirtualAddress) < 90380 <00080000+ 000261BC(Virtual Size).
- Therefore, it exists in .rdata section. PointerToRawData == 0x65000.
- FOA = 0x90380 - 0x80000 + 0x65000 == 0x75380

Export Directory Table

```
00075380  00 00 00 00 CF 77 02 59  00 00 00 00 6A 41 09 00
00075390  01 00 00 00 2D 06 00 00  2D 06 00 00 A8 03 09 00
000753A0  5C 1C 09 00 10 55 09 00  A0 62 01 00 A4 41 09 00
```

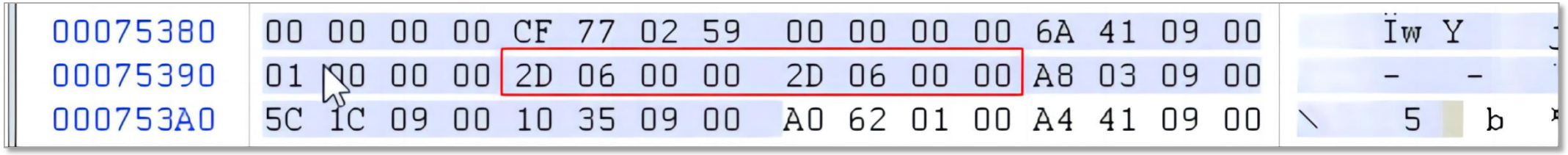
- **Name: starting at 13th bytes with value 0x9416A, which is a RVA. Its FOA == 7916A**

```
00079160  28 06 29 06 2A 06 2B 06  2C 06 4B 45 52 4E 45 4C  ( ) * + , KERNEL
00079170  33 32 2E 64 6C 6C 00 42  61 73 65 54 68 72 65 61  32.dll BaseThrea
```

Export Directory Table

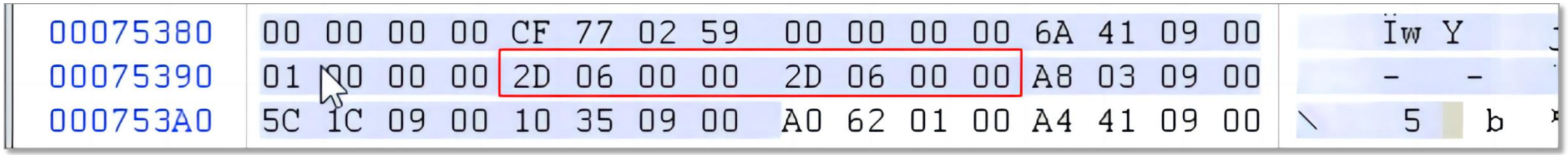
```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;            // Reserved, must be 0.
    DWORD   TimeDateStamp;              //The time and date that the export data was created.
    WORD    MajorVersion;               //The major version number.
    WORD    MinorVersion;               //The minor version number.
    DWORD   Name;                       //The address of the ASCII string that contains the name of the DLL.
    DWORD   Base;                       //The starting ordinal number for exports in this image.
                                        //It is usually set to 1.
    DWORD   NumberOfFunctions;          //The number of entries in the export address table.
    DWORD   NumberOfNames;              //The number of entries in the name pointer table.
    DWORD   AddressOfFunctions;         //The address of the export address table.
    DWORD   AddressOfNames;             //The address of the export name pointer table.
    DWORD   AddressOfNameOrdinals;      //The address of the ordinal table, relative to the image base.
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

NumberOfFunctions and NumberOfNames



- In this case, the number of all exported functions is 62d. The number of exported functions by name is 62d.
- The number of all exported functions - The number of exported functions by name == the number of differences.
- That means other functions may not be exported or exported by ordinal number.

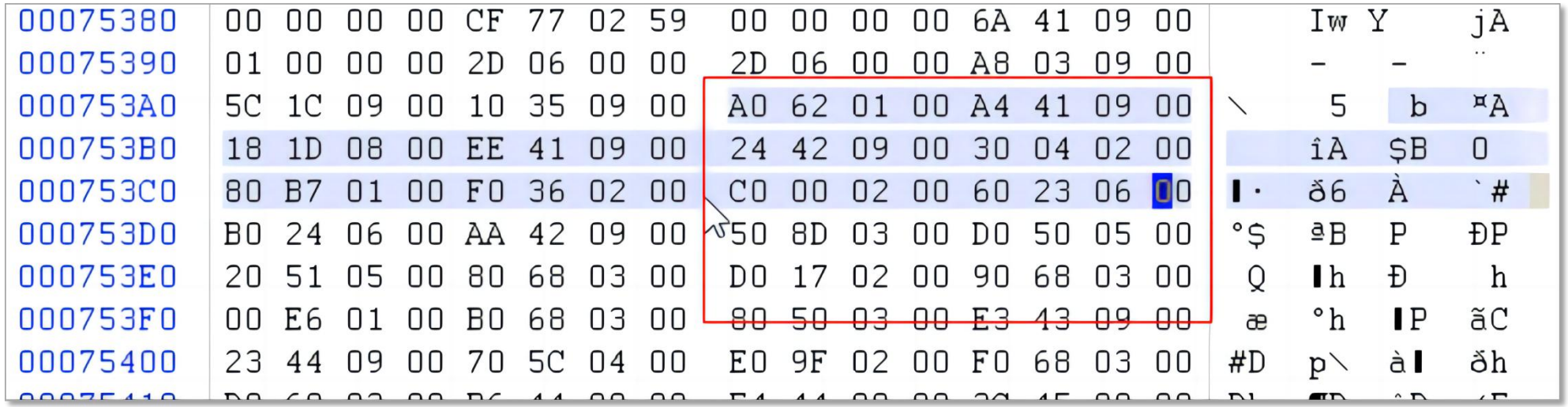
Export Directory Table



- Similarly, AddressOfFunctions FOA == 0x753A8
- AddressOfNames FOA == 0x76c5c
- AddressOfNameOrdinals FOA == 0x78510

AddressOfFunctions

AddressOfFunctions FOA == 0x753A8

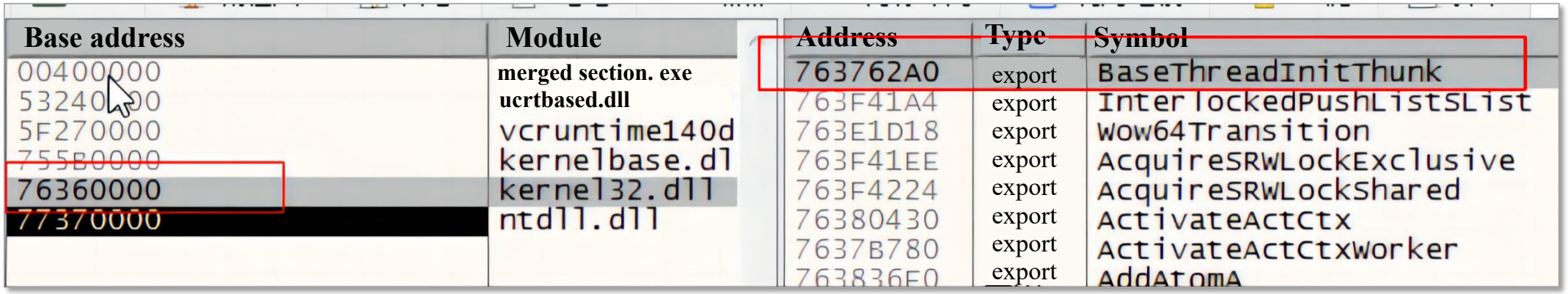


00075380	00	00	00	00	CF	77	02	59	00	00	00	00	6A	41	09	00
00075390	01	00	00	00	2D	06	00	00	2D	06	00	00	A8	03	09	00
000753A0	5C	1C	09	00	10	35	09	00	A0	62	01	00	A4	41	09	00
000753B0	18	1D	08	00	EE	41	09	00	24	42	09	00	30	04	02	00
000753C0	80	B7	01	00	F0	36	02	00	C0	00	02	00	60	23	06	00
000753D0	B0	24	06	00	AA	42	09	00	50	8D	03	00	D0	50	05	00
000753E0	20	51	05	00	80	68	03	00	D0	17	02	00	90	68	03	00
000753F0	00	E6	01	00	B0	68	03	00	80	50	03	00	E3	43	09	00
00075400	23	44	09	00	70	5C	04	00	E0	9F	02	00	F0	68	03	00

- In function address table, each entry is 4 bytes. The number of entries is determined by NumberOfFunctions.

Export Directory Table

- For example, the RVA of the first entry is 0x0162A0.
- We use the RVA of the first entry + ImageBase is the function address.

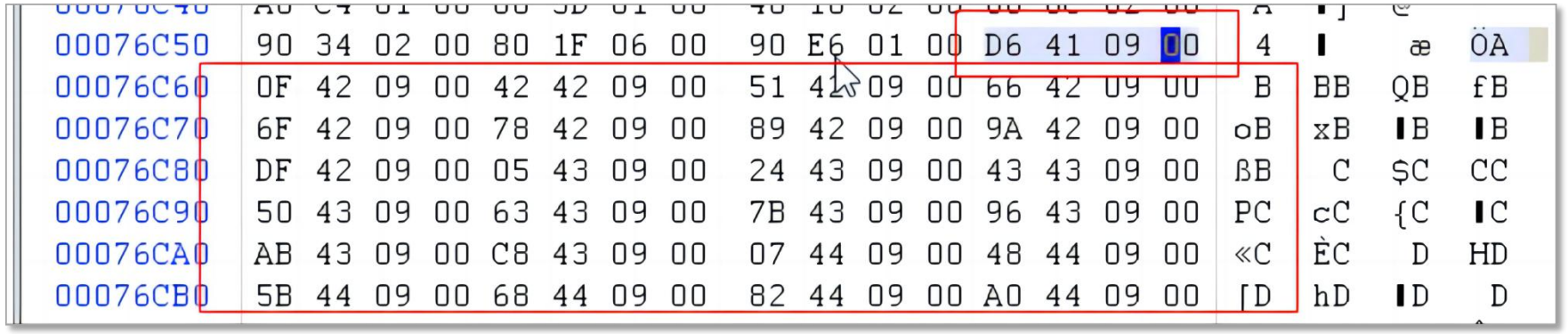


Base address	Module
00400000	merged section. exe
53240000	ucrtbased.dll
5F270000	vcruntime140d
755B0000	kernelbase.dl
76360000	kernel32.dll
77370000	ntdll.dll

Address	Type	Symbol
763762A0	export	BaseThreadInitThunk
763F41A4	export	InterlockedPushListSList
763E1D18	export	Wow64Transition
763F41EE	export	AcquireSRWLockExclusive
763F4224	export	AcquireSRWLockShared
76380430	export	ActivateActCtx
7637B780	export	ActivateActCtxWorker
763836E0	export	AddAtomA

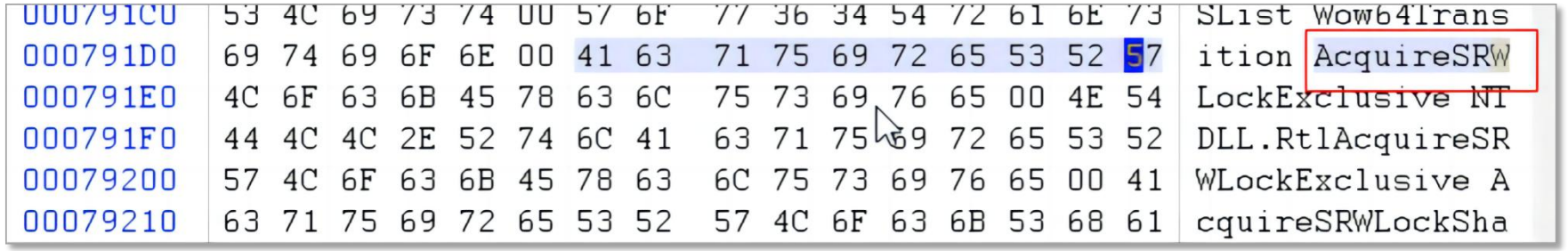
AddressOfNames

AddressOfNames FOA == 0x76c5c



```
00076C50  90 34 02 00 80 1F 06 00  90 E6 01 00 D6 41 09 00   4 ▌    æ  ÖA
00076C60  0F 42 09 00 42 42 09 00  51 42 09 00 66 42 09 00   B BB  QB  fB
00076C70  6F 42 09 00 78 42 09 00  89 42 09 00 9A 42 09 00  oB xB ▌B ▌B
00076C80  DF 42 09 00 05 43 09 00  24 43 09 00 43 43 09 00  ßB  C $C CC
00076C90  50 43 09 00 63 43 09 00  7B 43 09 00 96 43 09 00  PC cC {C ▌C
00076CA0  AB 43 09 00 C8 43 09 00  07 44 09 00 48 44 09 00  «C ÈC  D HD
00076CB0  5B 44 09 00 68 44 09 00  82 44 09 00 A0 44 09 00  [D hD ▌D  D
```

FOA = 0x941D6 - 0x80000 + 0x65000 = 0x791D6



```
000791C0  53 4C 69 73 74 00 57 6F  77 36 34 54 72 61 6E 73  SList Wow64Trans
000791D0  69 74 69 6F 6E 00 41 63  71 75 69 72 65 53 52 57  ition AcquireSRW
000791E0  4C 6F 63 6B 45 78 63 6C  75 73 69 76 65 00 4E 54  LockExclusive NT
000791F0  44 4C 4C 2E 52 74 6C 41  63 71 75 69 72 65 53 52  DLL.RtlAcquireSR
00079200  57 4C 6F 63 6B 45 78 63  6C 75 73 69 76 65 00 41  WLockExclusive A
00079210  63 71 75 69 72 65 53 52  57 4C 6F 63 6B 53 68 61  cquireSRWLockSha
```

AddressOfNameOrdinals

AddressOfNameOrdinals FOA == 0x78510

00078510	03	00	04	00	05	00	06	00	07	00	08	00	09	00	0A	00
00078520	0B	00	0C	00	0D	00	0E	00	0F	00	10	00	11	00	12	00
00078530	13	00	14	00	15	00	16	00	17	00	18	00	19	00	1A	00

Example

AddressOfFunctions	AddressOfNameOrdinals	AddressOfNames
0 0x1010 Sub	0 0x0100	0 Add
1 0x2020 Add	1 0x0000	1 Sub
2 0x3030 Div	2 0x0200	2 Div

Part Two

02

Relocation

2025-8-26

Relocation Table

```
Typedef struct _IMAGE_BASE_RELOCATION{
        DWORD VirtualAddress
        DWORD SizeOfBlock
}IMAGE_BASE_RELOCATION,*PIMAGE_BASE_RELOCATION;
```

PE

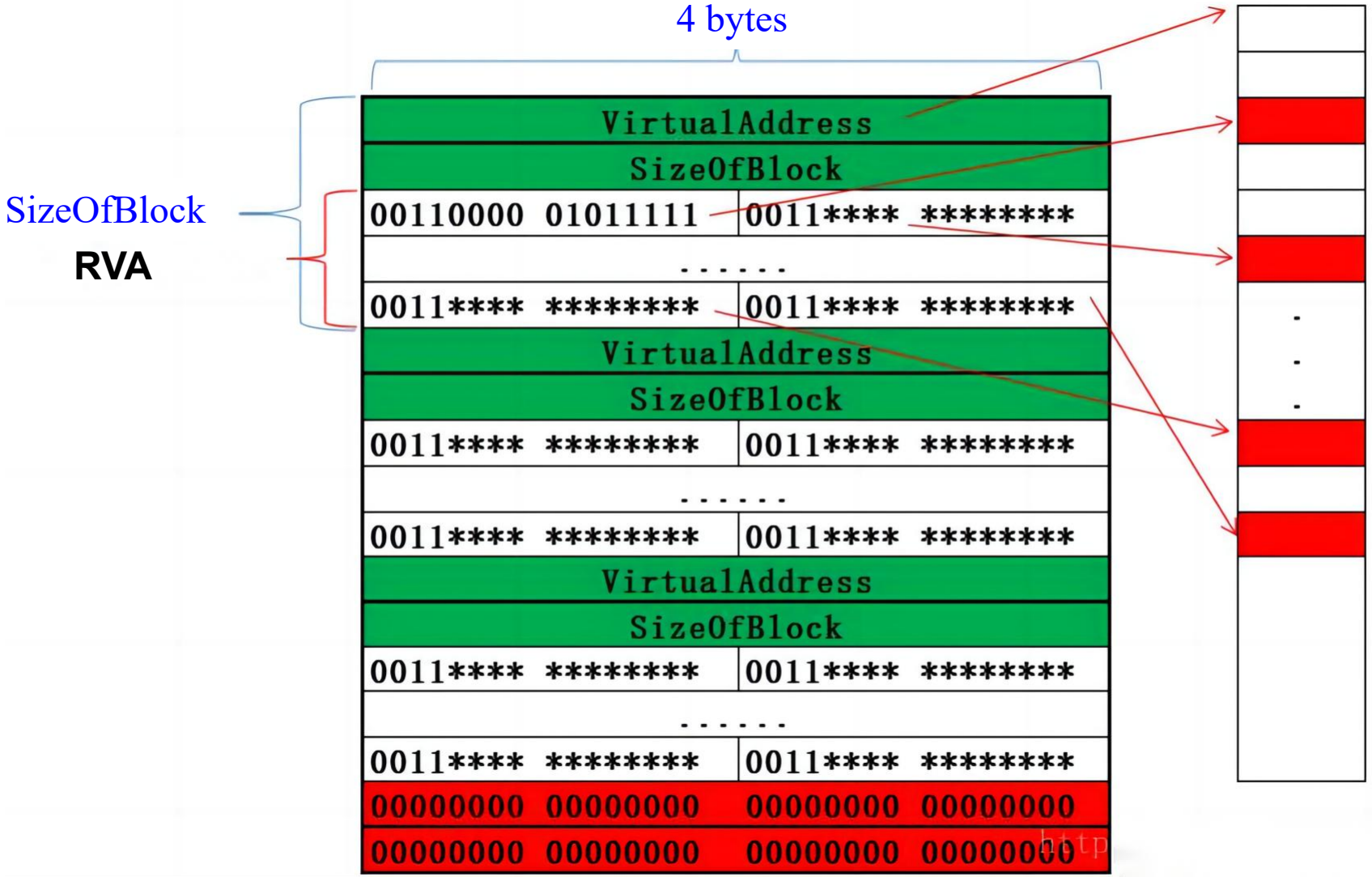
Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of Reloc.	Num. of Linenum.
.text	400	1C00	1000	1AC8	60000060	0	0	0
>	2000	^	2AC8	^	r-x			
.data	2000	200	3000	B0	C0000040	0	0	0
>	2200	^	30B0	^	rw-			
.rdata	2200	C00	4000	B00	40000040	0	0	0
.pdata	2E00	400	5000	240	40000040	0	0	0
.xdata	3200	200	6000	1B8	40000040	0	0	0
.bss	0	0	7000	180	C0000080	0	0	0
.idata	3400	800	8000	7DC	C0000040	0	0	0
.CRT	3C00	200	9000	60	C0000040	0	0	0
.tls	3E00	200	A000	10	C0000040	0	0	0
.rsrc	4000	600	B000	4E8	C0000040	0	0	0
.reloc	4600	200	C000	80	42000040	0	0	0

```
Typedef struct _IMAGE_BASE_RELOCATION{
DWORD VirtualAddress
DWORD SizeOfBlock
}IMAGE_BASE_RELOCATION,*PIMAGE_BASE_RELOCATION;
```

IMAGE_DIRECTORY_ENTRY_BASERELOC
DWORD VirtualAddress:0000 C000
DWORD Size:0000 0080

C000 – C000(VA of section)+4600(FOA of section)=4600

Base address: VirtualAddress



PE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4600	00	20	00	00	0C	00	00	00	A8	AA	00	00	00	30	00	00
4610	10	00	00	00	10	A0	60	A0	70	A0	80	A0	90	A0	98	A0
4620	A0	A0	00	00	00	40	00	00	4C	00	00	00	40	A0	60	A0
4630	68	A0	70	A0	78	A0	70	A3	80	A3	90	A3	A0	A3	B0	A3
4640	C0	A3	D0	A3	E0	A3	F0	A3	00	A4	10	A4	20	A4	30	A4
4650	40	A4	50	A4	60	A4	70	A4	80	A4	90	A4	A0	A4	B0	A4
4660	C0	A4														
4670	00	90														
4680	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4690	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
46A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
46B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
46C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
46D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
46E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Part Three

03

Retrofitting

Retrofiting

Malware retrofitting refers to the process of modifying or updating existing malware to enhance its functionality, stealthiness, or evasion capabilities. This practice is typically carried out by cybercriminals or hackers to adapt their malicious code to changing security measures, making it more difficult for antivirus products to detect and remove the malware.

Retrofiting Technique

Here are some common objectives and techniques associated with malware retrofitting:

- **Evasion of Antivirus and Security Software:** Malware authors often retrofit their code to evade detection by antivirus and security software. This may involve altering the code's structure, changing its signatures, or using polymorphic techniques to generate new variants that appear different to security scanners.

- **Persistence:** Malware often seeks to maintain a persistent presence on an infected system. Retrofitting may involve enhancing the malware's ability to survive system reboots, updates, or antivirus scans.

Retrofiting Technique

Here are some common objectives and techniques associated with malware retrofitting:

- **Payload Delivery:** Malware may be retrofitted to deliver additional payloads or modules. For example, a Trojan horse may be updated to download and execute other malicious software, such as ransomware or keyloggers.

- **Data Exfiltration:** Retrofitting can add data exfiltration capabilities to malware. This enables it to steal sensitive information from infected systems and transmit it to command and control servers controlled by cybercriminals.

Retrofiting Technique

Here are some common objectives and techniques associated with malware retrofitting:

- **Evasion of Sandboxing and Analysis:** Malware retrofitting may include techniques to detect if it is running in a sandbox or virtualized environment used for security analysis. If detected, the malware may behave differently or remain dormant to avoid detection.

- **Dynamic Command and Control:** Retrofitting can enhance the malware's ability to communicate with command and control servers dynamically. This makes it harder for security researchers to track and disrupt malicious networks.

Retrofiting Technique

Here are some common objectives and techniques associated with malware retrofitting:

- **Rootkit Functionality:** Some malware is retrofitted to include rootkit capabilities, allowing it to gain elevated privileges and hide from system monitoring tools.

- **Obfuscation:** Code obfuscation techniques may be applied during retrofitting to make the malware's code more challenging to analyze and reverse-engineer.

Conclusion

- It's important to note that malware retrofitting is an ongoing arms race between cybercriminals and cybersecurity professionals. As security measures improve, malware authors adapt their code to counteract these defenses.
- Consequently, cybersecurity experts continuously work to develop better detection and prevention techniques to combat evolving malware threats.

Project3

You should develop program2.c that has to be implemented with the following functionality:

- print a string "hello program2".
- Read encrypted program1 at the last section of program2.exe.
- Decrypt it to get the original program1.exe.
- Create the process in suspended form by using API "CreateProcess", the process to be created is program2.exe.
- Get the context of the program2.exe (ImageBase and OEP).
- Uninstaller (NtUnmapViewOfSection).
- Allocate space (by using API "VirtualAllocEx") at the specified location which is "ImageBase" of program1.exe, and the size is the SizeOfImage of program1.exe.
- If the application space is successful, stretch the program1.exe and copy it to the space (by using WriteProcessMemory).
- If the application space fails, but there is a relocation table, apply for space at any position, then stretch, copy, and repair the relocation table of the program1.exe.
- Modify the Context of the program. Change the ImageBase of the Context of the program2.exe to the ImageBase of program1.exe and change the OEP of the Context of the program2.exe to the OEP of program1.exe.
- Set the Context and restore the main thread
- The replacement is successful

Project3

The source code of program1.exe is to print a string "hello program1".

- You have to develop program1.c that print the string.
- Compile program1.c to produce program1.exe.

- We have to develop a program3.c that first encrypt the virus (in our case, it is program1.exe) by XORing it with 0x40 and then attached the encrypted virus to the end of program2.exe. After it is compiled, producing program3.exe.
- After running program3.exe, you will get the new program2.exe that has the encrypted program1.exe at its last section.
- If we run the new program2.exe, it first prints "hello program2" and then prints "hello program1".

MONTANA STATE UNIVERSITY

THE END

Fangtian Zhong

CSCI 591

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

09/16/2025