PE Binary Infection

Maycon Maia Vitali aka 0ut0fBound

Agenda

- Introdução
- Estrutura do PE
  - Headers / Directories / Sections
- RVA (Relative Virtual Address) e Align
  - Rva2Section() / Rva2Offset()
  - Alinhando por Arquivo e por Endereço Virtual
- Processo de Infecçao
  - Adicionando nova Section Header
  - Definindo a Section Data
  - Arredondando o novo Binário
- Ferramentas Auxiliares
- Conclusão

Introdução

- PE = Portable Executable
- Modificada versão do COFF
- Introduzido inicialmente no Microsoft Windows NT 3.1
- Utilizado em EXE, OBJ, DLL e SYS

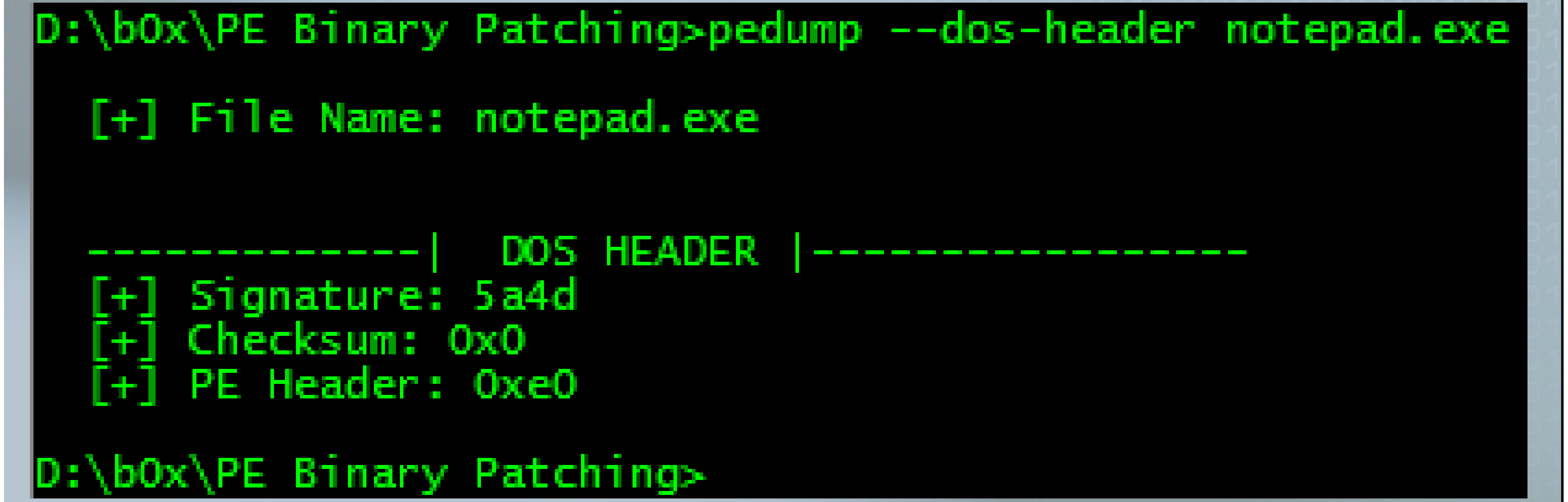
Estrutura do arquivo PE

MS-DOS INFORMATION	IMAGE_DOS_HEADER
	MS-DOS STUB PROGRAM
IMAGE_NT_HEADER	IMAGE_FILE_HEADER
	IMAGE_OPTIONAL_HEADER32
SECTION HEADERS	IMAGE_SECTION_HEADER[0]
	...
	IMAGE_SECTION_HEADER[N]
SECTIONS DATA	SECTION[0]
	...
	SECTION[N]

IMAGE_DOS_HEADER

```
typedef struct _IMAGE_DOS_HEADER {
    WORD e_magic; //- "MZ"
    WORD e_cblp;
    WORD e_cp;
    WORD e_crlc;
    WORD e_cparhdr;
    WORD e_minalloc;
    WORD e_maxalloc;
    WORD e_ss;
    WORD e_sp;
    WORD e_csum;
    WORD e_ip;
    WORD e_cs;
    WORD e_lfarlc;
    WORD e_ovno;
    WORD e_res[4];
    WORD e_oemid;
    WORD e_oeminfo;
    WORD e_res2[10];
    LONG e_lfanew;
} IMAGE_DOS_HEADER,*PIMAGE_DOS_HEADER;
```

IMAGE_DOS_HEADER



```
D:\b0x\PE Binary Patching>pedump --dos-header notepad.exe

  [+] File Name: notepad.exe


  -------------|  DOS HEADER |------------------
  [+] Signature: 5a4d
  [+] Checksum: 0x0
  [+] PE Header: 0xe0

D:\b0x\PE Binary Patching>
```

MS-DOS Stub Program

```
0000 | PUSH   CS
0001 | POP    DS
0002 | MOV    DX, 000E
0005 | MOV    AH, 09      ; write()
0007 | INT    21
0009 | MOV    AX, 4C01    ; exit()
000C | INT    21
000E | DB     "This program cannot be run “
     |        “in DOS mode.”,13,10,”$”
```

IMAGE_NT_HEADER

```
typedef struct _IMAGE_NT_HEADERS {
        DWORD Signature;
        IMAGE_FILE_HEADER FileHeader;
        IMAGE_OPTIONAL_HEADER OptionalHeader;
} IMAGE_NT_HEADERS,*PIMAGE_NT_HEADERS;
```

Assinatura **0x4550** identifica um arquivo PE.

IMAGE_FILE_HEADER

```
typedef struct _IMAGE_FILE_HEADER {
        WORD Machine;
        WORD NumberOfSections;
        DWORD TimeDateStamp;
        DWORD PointerToSymbolTable;
        DWORD NumberOfSymbols;
        WORD SizeOfOptionalHeader;
        WORD Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

IMAGE_FILE_HEADER

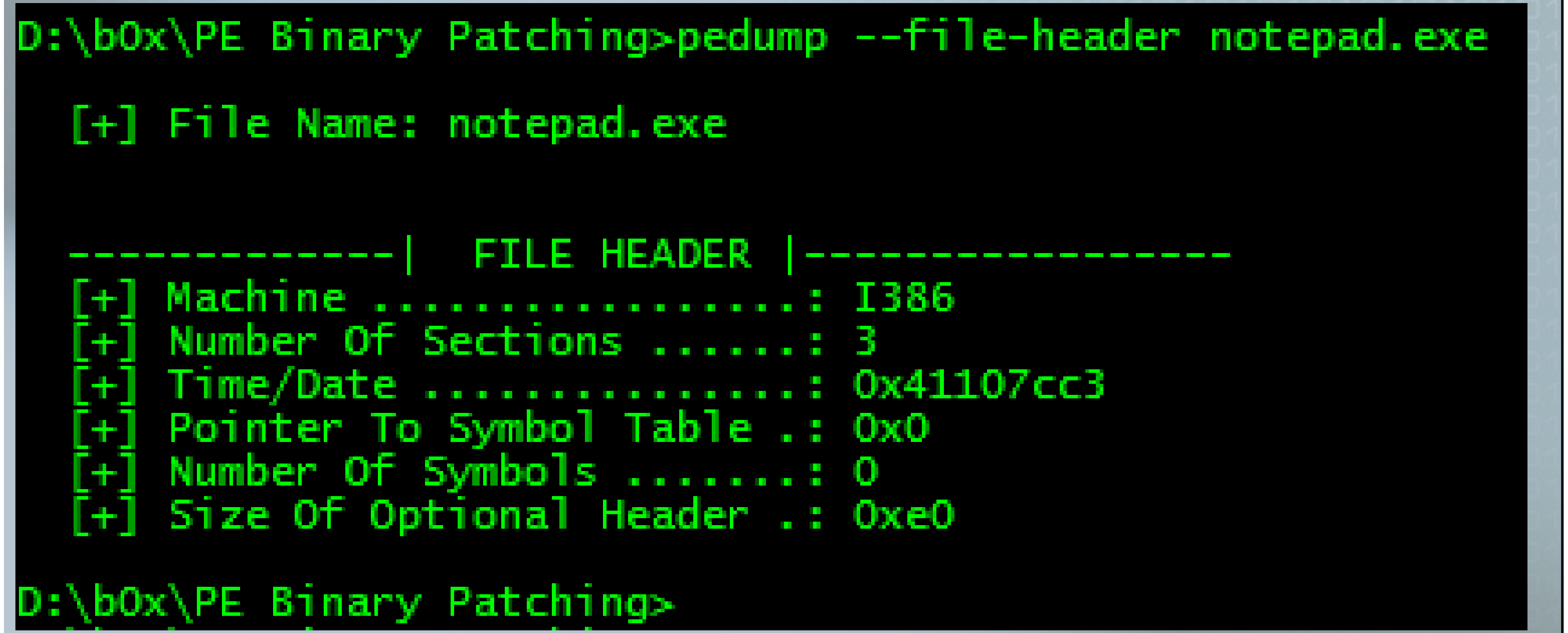
IMAGE_FILE_HEADER->Machine

```
#define IMAGE_FILE_MACHINE_UNKNOWN   0
#define IMAGE_FILE_MACHINE_I386      332
#define IMAGE_FILE_MACHINE_R3000     354
#define IMAGE_FILE_MACHINE_R4000     358
#define IMAGE_FILE_MACHINE_R10000    360
#define IMAGE_FILE_MACHINE_ALPHA     388
#define IMAGE_FILE_MACHINE_POWERPC   496
```

IMAGE_FILE_HEADER->Characteristics

```
#define IMAGE_FILE_EXECUTABLE_IMAGE      2
#define IMAGE_FILE_DLL                   8192
```

IMAGE_FILE_HEADER



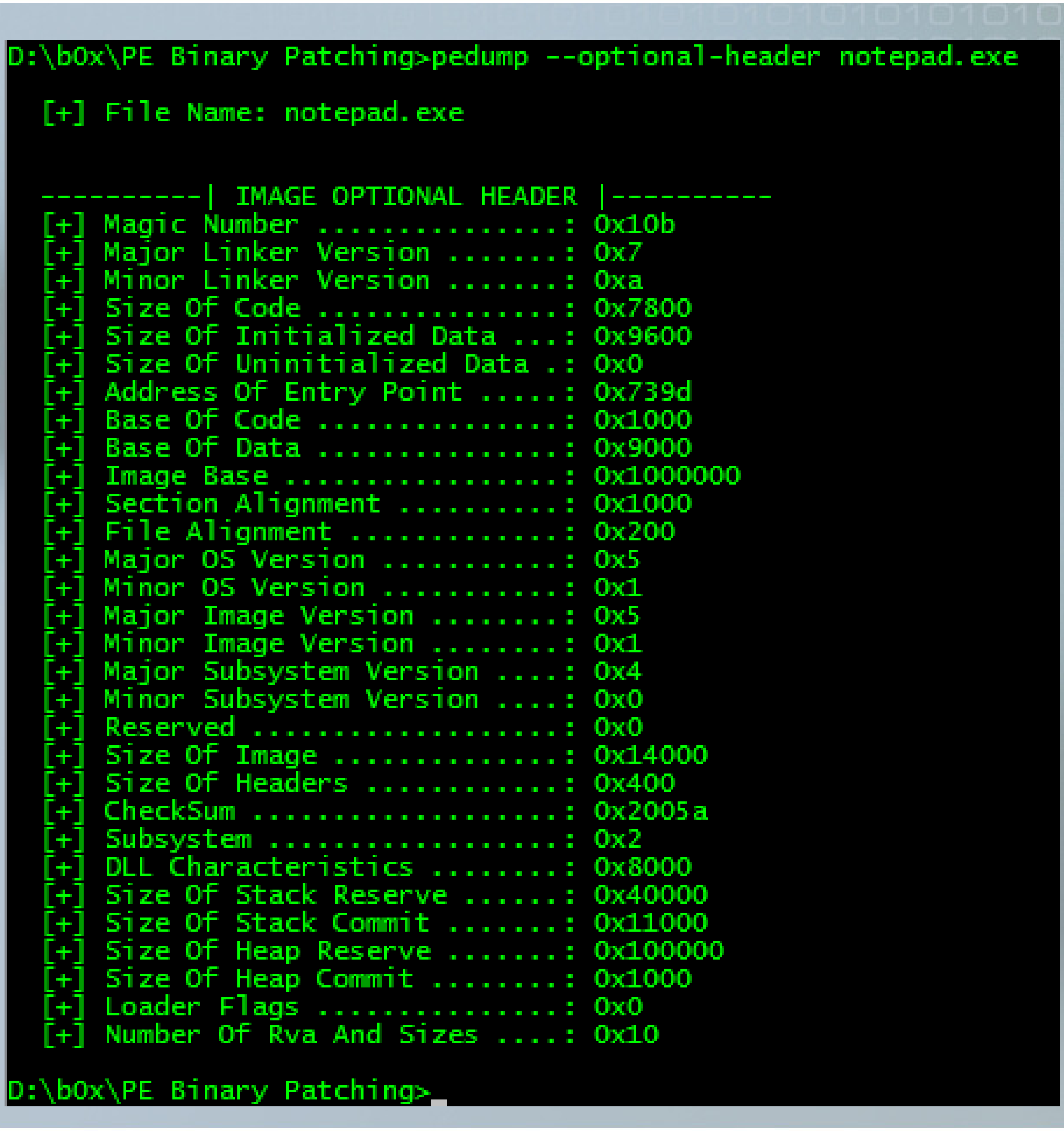
```
D:\b0x\PE Binary Patching>pedump --file-header notepad.exe

   [+] File Name: notepad.exe


   --------------|  FILE HEADER |------------------
   [+] Machine ..................: I386
   [+] Number Of Sections .......: 3
   [+] Time/Date ................: 0x41107cc3
   [+] Pointer To Symbol Table .: 0x0
   [+] Number Of Symbols .......: 0
   [+] Size Of Optional Header .: 0xe0

D:\b0x\PE Binary Patching>
```

IMAGE_OPTIONAL_HEADER32

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    ...
    DWORD AddressOfEntryPoint;
    ...
    DWORD ImageBase;
    DWORD SectionAlignment;
    DWORD FileAlignment;
    ...
    DWORD SizeOfImage;
    ...
    IMAGE_DATA_DIRECTORY
    DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER,*PIMAGE_OPTIONAL_HEADER;
```

```
#define IMAGE_DIRECTORY_ENTRY_EXPORT      0
#define IMAGE_DIRECTORY_ENTRY_IMPORT      1
#define IMAGE_DIRECTORY_ENTRY_RESOURCE  2
#define IMAGE_DIRECTORY_ENTRY_COPYRIGHT 7
```

IMAGE_OPTIONAL_HEADER32



```
D:\b0x\PE Binary Patching>pedump --optional-header notepad.exe

  [+] File Name: notepad.exe


  ----------| IMAGE OPTIONAL HEADER |----------
  [+] Magic Number ................: 0x10b
  [+] Major Linker Version ........: 0x7
  [+] Minor Linker Version ........: 0xa
  [+] Size Of Code ................: 0x7800
  [+] Size Of Initialized Data ....: 0x9600
  [+] Size Of Uninitialized Data ..: 0x0
  [+] Address Of Entry Point ......: 0x739d
  [+] Base Of Code ................: 0x1000
  [+] Base Of Data ................: 0x9000
  [+] Image Base ..................: 0x1000000
  [+] Section Alignment ...........: 0x1000
  [+] File Alignment ..............: 0x200
  [+] Major OS Version ............: 0x5
  [+] Minor OS Version ............: 0x1
  [+] Major Image Version .........: 0x5
  [+] Minor Image Version .........: 0x1
  [+] Major Subsystem Version .....: 0x4
  [+] Minor Subsystem Version .....: 0x0
  [+] Reserved ....................: 0x0
  [+] Size Of Image ...............: 0x14000
  [+] Size Of Headers .............: 0x400
  [+] CheckSum ....................: 0x2005a
  [+] Subsystem ...................: 0x2
  [+] DLL Characteristics .........: 0x8000
  [+] Size Of Stack Reserve .......: 0x40000
  [+] Size Of Stack Commit ........: 0x11000
  [+] Size Of Heap Reserve ........: 0x100000
  [+] Size Of Heap Commit .........: 0x1000
  [+] Loader Flags ................: 0x0
  [+] Number Of Rva And Sizes .....: 0x10

D:\b0x\PE Binary Patching>
```

DataDirectory[IMAGE_DIRECTORY_ENTRY_IMPORT]

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    _ANONYMOUS_UNION union {
            DWORD Characteristics;
            DWORD OriginalFirstThunk;
    } DUMMYUNIONNAME;
    DWORD TimeDateStamp;
    DWORD ForwarderChain;
    DWORD Name;
    DWORD FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR,*PIMAGE_IMPORT_DESCRIPTOR;
```

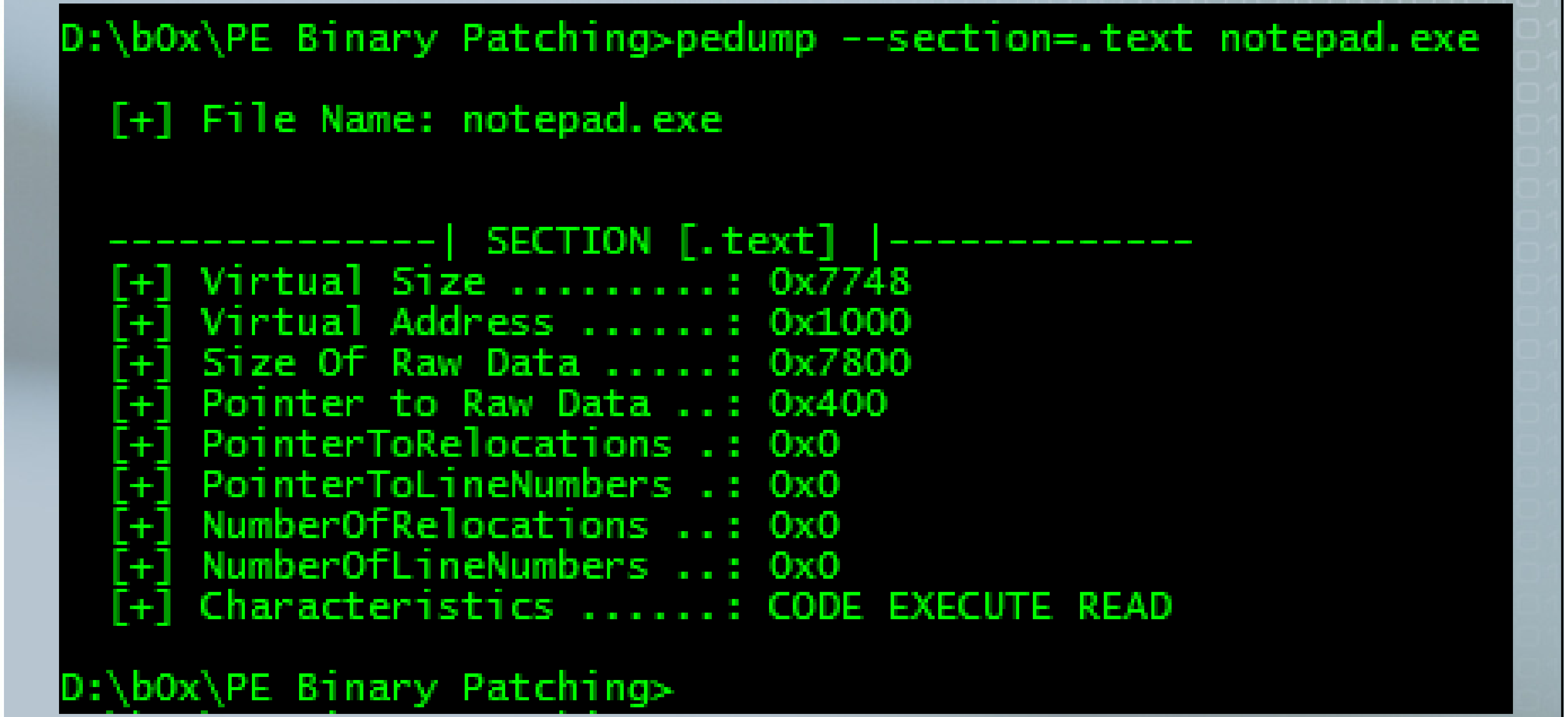
Name: Kernel32.dll
FirstThunk: (IMAGE_IMPORT_BY_NAME)

```
typedef struct _IMAGE_IMPORT_BY_NAME {
            WORD Hint;
            BYTE Name[1];
} IMAGE_IMPORT_BY_NAME,*PIMAGE_IMPORT_BY_NAME;
```

IMAGE_SECTION_HEADER

```
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;
```

IMAGE_SECTION_HEADER



```
D:\b0x\PE Binary Patching>pedump --section=.text notepad.exe

  [+] File Name: notepad.exe


  ---------------| SECTION [.text] |--------------
  [+] Virtual Size ..........: 0x7748
  [+] Virtual Address .......: 0x1000
  [+] Size Of Raw Data ......: 0x7800
  [+] Pointer to Raw Data ...: 0x400
  [+] PointerToRelocations .: 0x0
  [+] PointerToLineNumbers .: 0x0
  [+] NumberOfRelocations ..: 0x0
  [+] NumberOfLineNumbers ..: 0x0
  [+] Characteristics .......: CODE EXECUTE READ

D:\b0x\PE Binary Patching>
```

PE_Section()

```
PIMAGE_SECTION_HEADER PE_Section(unsigned int n, char *cFileBuffer) {

    PIMAGE_DOS_HEADER hdrDOS;
    PIMAGE_NT_HEADERS hdrNT;

    DWORD nSectionPosition;

    hdrDOS = (PIMAGE_DOS_HEADER)cFileBuffer;
    hdrNT  = (PIMAGE_NT_HEADERS)((DWORD)cFileBuffer + hdrDOS->e_lfanew - 1);

    nSectionPosition = hdrDOS->e_lfanew                          /* Start of PE Header */
                   + 4                                           /* Sizeof signature */
                   + IMAGE_SIZEOF_FILE_HEADER
                   + hdrNT->FileHeader.SizeOfOptionalHeader
                   + (n * IMAGE_SIZEOF_SECTION_HEADER); /* Calcule the section position */

    return (PIMAGE_SECTION_HEADER)((DWORD)cFileBuffer + nSectionPosition);

}
```

Rva2Section()

```
PIMAGE_SECTION_HEADER Rva2Section(DWORD nRvaAddress, char *cFileBuffer) {
    PIMAGE_DOS_HEADER hdrDOS;
    PIMAGE_NT_HEADERS hdrNT;
    PIMAGE_SECTION_HEADER hdrSection;

    unsigned int nCount;
    DWORD nSectionPosition;

    hdrDOS = (PIMAGE_DOS_HEADER)cFileBuffer;
    hdrNT  = (PIMAGE_NT_HEADERS)((DWORD)cFileBuffer + hdrDOS->e_lfanew - 1);

    for (nCount = 0; nCount <  hdrNT->FileHeader.NumberOfSections; nCount++) {

        hdrSection = (PIMAGE_SECTION_HEADER)PE_Section(nCount, cFileBuffer);

        if (
            (nRvaAddress >= hdrSection->VirtualAddress) &&
            (nRvaAddress <  hdrSection->VirtualAddress + hdrSection->SizeOfRawData)
        ) return hdrSection;

    }

    return NULL;
}
```

Rva2Offset()

```
PIMAGE_SECTION_HEADER Offset2Section(DWORD nOffsetAddress, char *cFileBuffer) {
    PIMAGE_DOS_HEADER hdrDOS;
    PIMAGE_NT_HEADERS hdrNT;
    PIMAGE_SECTION_HEADER hdrSection;

    unsigned int nCount;
    unsigned long nSectionPosition;

    hdrDOS = (PIMAGE_DOS_HEADER)cFileBuffer;
    hdrNT  = (PIMAGE_NT_HEADERS)((DWORD)cFileBuffer + hdrDOS->e_lfanew - 1);

    for (nCount = 0; nCount <  hdrNT->FileHeader.NumberOfSections; nCount++) {

                hdrSection = (PIMAGE_SECTION_HEADER)PE_Section(nCount, cFileBuffer);

        if (
            (nOffsetAddress >= hdrSection->PointerToRawData) &&
            (nOffsetAddress <  hdrSection->PointerToRawData + hdrSection->SizeOfRawData
        )) return hdrSection;
    }

    return NULL;
}
```

PE_MakeAlign()

```
DWORD PE_MakeAlign(DWORD nValue, DWORD nBaseAlign) {
    return ((nValue + nBaseAlign - 1)/nBaseAlign)*nBaseAlign;
}
```

- OptionalHeader.SectionAlignment
  - SectionHeader.VirtualAddress
  - SectionHedar.Misc.VirtualSize

- OptionalHeader.FileAlignment
  - SectionHeader.PointerToRawData
  - SectionHeader.SizeOfRawData

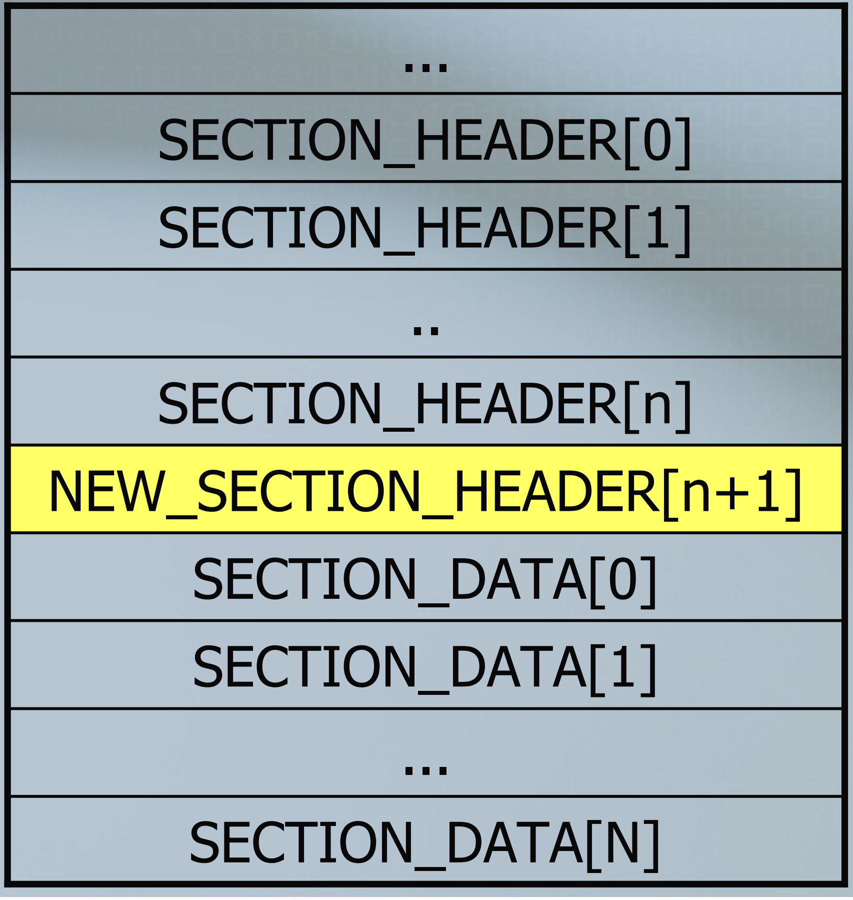
Processo de Infecção

Alterando o AddressOfEntryPoint

- Leia o IMAGE_OPTINAL_HEADER32
- Mude o valor do campo AddressOfEntryPoint
- Sobrescreva o cabeçalho

Adicionando IMAGE_SECTION_HEADER

Verificar se possui espaço entre o final da ultima SECTION e o inicio dos dados da primeira SECTION.



...

SECTION_HEADER[0]

SECTION_HEADER[1]

..

SECTION_HEADER[n]

NEW_SECTION_HEADER[n+1]

SECTION_DATA[0]

SECTION_DATA[1]

...

SECTION_DATA[N]

Definindo a SECTION data

- Windows Shellcode
- Buscando pela Import Table
- Posicionar no final do arquivo? Não!
- Procurar a última SECTION e calcular o final de seus dados no arquivo:
  - SECTION. PointerToRawData + SECTION. SizeOfRawData

Gravando SECTION Data

- Como nova SECTION criada
- Procurar buracos no código
- Sobrescrita de código

Arredondando o novo binário

- Atualizando o AddressOfEntryPoint e SizeOfImage

```
Read (OptionalHeader);
OptionalHeader.AddressOfEntryPoint =
   NovaSection.VirtualAddress;
OptionalHeader.SizeOfImage =
   NovaSection.VirtualAddress+NovaSection.VirtualSize
Write (OptionaHeader);
```

Ferramentas Auxiliares

- Explore Suite
  - http://www.ntcore.com
- OllyDbg
  - http://home.t-online.de/home/Ollydbg

Sem Dúvidas!!!

mayconmaia@yahoo.com.br
0ut0fBound old Chuck_Newbie