武汉大学国家网络安全学院
2022-2023 学年度第 二 学期
《计算机病毒》期末考试试卷 A 卷(开 卷)

专业：

	一					
分值	35					100

一. 计算题（共 3 小题，共 35 分）

1. 图 1 是某 MBR 分区格式硬盘的分区表信息，请问：
   （1） 该磁盘包含哪几种类型的分区？请说明理由。（4 分）
   （2） 请给出各分区起始和结束扇区位置，以及分区的大小（给出计算公式即可）。（6 分）

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000001B0	00	00	00	00	00	2C	44	63	6E	D0	6E	D0	00	00	80	01
0000001C0	01	00	07	FE	FF	FF	3F	00	00	00	00	00	C0	03	00	FE
0000001D0	FF	FF	0F	FE	FF	FF	43	00	C0	03	00	00	80	03	00	00
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA

图 1 某 MBR 分区格式硬盘的分区表信息

2. 下表是某 PE 文件的导出目录表的部分信息，PE 文件导出目录表的数据结构见附录。要求：
   （1） 给出下表中“请补全 1”~“请补全 4”这四个位置处的值。（每项 2 分）
   （2） 导出函数地址表在该 PE 文件上的偏移。（2 分）

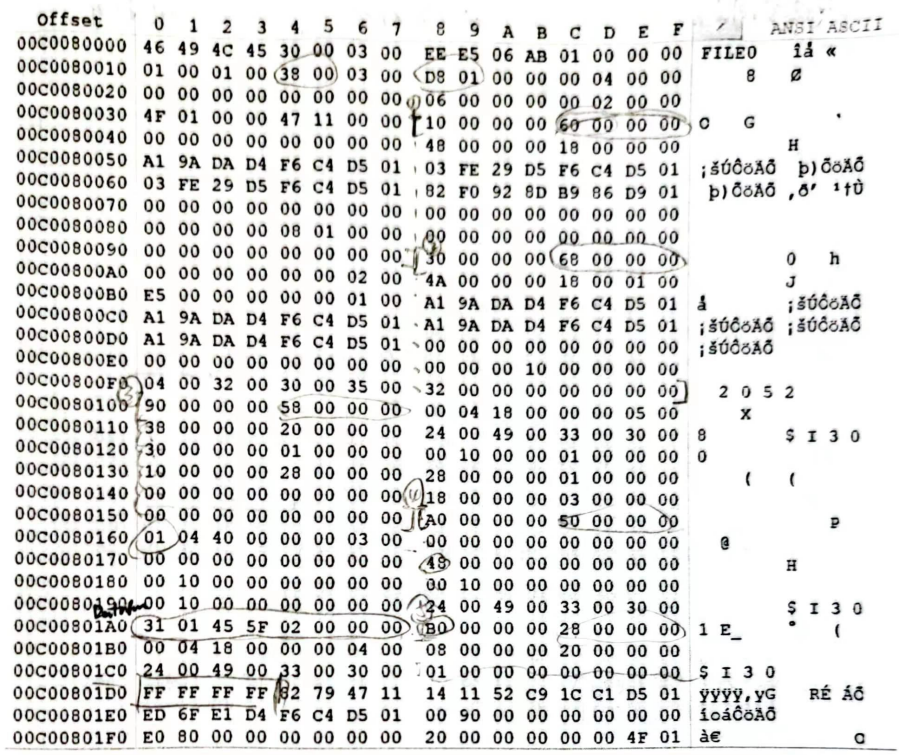
Offset（文件偏移）	Name	Value（十六进制）
F1A60	Characteristics	0
F1A64	TimeDateStamp	CD271E78
F1A68	MajorVersion	0
F1A6A	MinorVersion	0
F1A6C	Name	F9318
F1A70	Base	5
F1A74（请补全 1）	NumberOfFunctions	1B1B
F1A78	NumberOfNames	6
F1A7C	AddressOfFunctions	（请补全 3）
F1A80	AddressOfNames	F92F4
F1A84（请补全 2）	AddressOfNameOrdinals	（请补全 4）

3. 图3是一个实际NTFS文件系统中的某个文件记录（FR），请回答：

（1） 该FR中实际数据的起止偏移。（3分）

（2） 该FR中各属性的起止偏移。（6分）

（3） 找出该FR中一个非常驻属性，找到并解析其中的DataRun List。（6分）



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00C0080000	46	49	4C	45	30	00	03	00	EE	E5	06	AB	01	00	00	00	FILE0 îå «
00C0080010	01	00	01	00	38	00	03	00	D8	01	00	00	00	04	00	00	8 Ø
00C0080020	00	00	00	00	00	00	00	00	06	00	00	00	00	02	00	00	
00C0080030	4F	01	00	00	47	11	00	00	10	00	00	00	60	00	00	00	O G `
00C0080040	00	00	00	00	00	00	00	00	48	00	00	00	18	00	00	00	H
00C0080050	A1	9A	DA	D4	F6	C4	D5	01	03	FE	29	D5	F6	C4	D5	01	¡šÚÔöÄÕ þ)ÕöÄÕ
00C0080060	03	FE	29	D5	F6	C4	D5	01	82	F0	92	8D	B9	86	D9	01	þ)ÕöÄÕ ‚ð' ¹†Ù
00C0080070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00C0080080	00	00	00	00	08	01	00	00	00	00	00	00	00	00	00	00	
00C0080090	00	00	00	00	00	00	00	00	30	00	00	00	68	00	00	00	0 h
00C00800A0	00	00	00	00	00	00	02	00	4A	00	00	00	18	00	01	00	J
00C00800B0	E5	00	00	00	00	00	01	00	A1	9A	DA	D4	F6	C4	D5	01	å ¡šÚÔöÄÕ
00C00800C0	A1	9A	DA	D4	F6	C4	D5	01	A1	9A	DA	D4	F6	C4	D5	01	¡šÚÔöÄÕ ¡šÚÔöÄÕ
00C00800D0	A1	9A	DA	D4	F6	C4	D5	01	00	00	00	00	00	00	00	00	¡šÚÔöÄÕ
00C00800E0	00	00	00	00	00	00	00	00	00	00	00	10	00	00	00	00	
00C00800F0	04	00	32	00	30	00	35	00	32	00	00	00	00	00	00	00	2 0 5 2
00C0080100	90	00	00	00	58	00	00	00	00	04	18	00	00	00	05	00	X
00C0080110	38	00	00	00	20	00	00	00	24	00	49	00	33	00	30	00	8 $ I 3 0
00C0080120	30	00	00	00	01	00	00	00	00	10	00	00	01	00	00	00	0
00C0080130	10	00	00	00	28	00	00	00	28	00	00	00	01	00	00	00	((
00C0080140	00	00	00	00	00	00	00	00	18	00	00	00	03	00	00	00	
00C0080150	00	00	00	00	00	00	00	00	A0	00	00	00	50	00	00	00	P
00C0080160	01	04	40	00	00	00	03	00	00	00	00	00	00	00	00	00	@
00C0080170	00	00	00	00	00	00	00	00	48	00	00	00	00	00	00	00	H
00C0080180	00	10	00	00	00	00	00	00	00	10	00	00	00	00	00	00	
00C0080190	00	10	00	00	00	00	00	00	24	00	49	00	33	00	30	00	$ I 3 0
00C00801A0	31	01	45	5F	02	00	00	00	B0	00	00	00	28	00	00	00	1 E_ ° (
00C00801B0	00	04	18	00	00	00	04	00	08	00	00	00	20	00	00	00	
00C00801C0	24	00	49	00	33	00	30	00	01	00	00	00	00	00	00	00	$ I 3 0
00C00801D0	FF	FF	FF	FF	82	79	47	11	14	11	52	C9	1C	C1	D5	01	ÿÿÿÿ‚yG RÉ ÁÕ
00C00801E0	ED	6F	E1	D4	F6	C4	D5	01	00	90	00	00	00	00	00	00	íoáÔöÄÕ
00C00801F0	E0	80	00	00	00	00	00	00	20	00	00	00	00	00	4F	01	à€ O

图2 NTFS文件系统中的某个文件记录

二. 简答与论述题（共4小题，每题10分，共40分）

1. 在恶意代码检测方法中，基于特征码的检测与基于校验和的检测是两种常见方法，请比较它们的异同点和优缺点？
2. 请比较网络蠕虫与狭义计算机病毒这两类恶意代码的异同点。
3. 对远控木马而言，攻击者与木马之间有必要建立可靠的通信。请介绍和比较至少两种常见的木马通信方案。
4. 请结合“震网”病毒攻击事件，谈谈你对加强关键基础设施恶意代码防护的认识。

三. 代码分析题（共1小题，共25分）

1. 表1所列的C语言程序代码取自一份恶意代码设计报告，它演示了一种针对PE格式可执行文件的恶意代码植入和执行方案，为简洁起见，其中略去了所有错误处理代码。请认真阅读代码后回答：

（1）函数DoSomethingA的主要功能是什么？（4分）

（2）函数DoSomethingB的主要功能是什么？（5分）

（3）第 29 行代码中的 0x58 代表什么？（3 分）

（4）第 31 行代码的作用如注释（第 30 行）所述，请问为什么要做这一步？（4 分）

（5）请阐述表 1 代码所用的恶意代码植入方案和触发思路。（9 分）

【提示：代码中用到的主要数据结构和函数，其说明见本试卷附录。】

表 1 某恶意代码设计报告中的部分演示代码

```
1.  void DoSomethingA(const char *targetPE, unsigned char *payload, int plSize)
2.  {
3.      HANDLE hUpdate = NULL;
4.      hUpdate = BeginUpdateResource(targetPE, TRUE);
5.      UpdateResource(hUpdate, RT_BITMAP, MAKEINTRESOURCE(RT_BITMAP), 0, payload, plSize);
6.      EndUpdateResource(hUpdate, FALSE);
7.  }
8.
9.  DWORD_PTR DoSomethingB(const char *targetPE)
10. {
11.     // targetPE 为一可执行程序，假设此处已创建其进程，其进程句柄和基址分别为 hProcess 和 ImageBase。
12.     SIZE_T cbRead = 0;
13.     IMAGE_DOS_HEADER dosHeader = { 0 };
14.     IMAGE_NT_HEADERS ntHeader = { 0 };
15.     IMAGE_SECTION_HEADER sectionHeader = { 0 };
16.     ReadProcessMemory(hProcess, (LPCVOID)ImageBase, &dosHeader, sizeof(IMAGE_DOS_HEADER), &cbRead);
17.
18.     DWORD_PTR ntHeaderVA = ((DWORD_PTR)ImageBase + dosHeader.e_lfanew);
19.     ReadProcessMemory(hProcess, (LPCVOID)ntHeaderVA, &ntHeader, sizeof(IMAGE_NT_HEADERS), &cbRead);
20.
21.     DWORD_PTR sectAddr = ntHeaderVA + (DWORD_PTR)(sizeof(DWORD) + sizeof(IMAGE_FILE_HEADER)
22.                         + ntHeader.FileHeader.SizeOfOptionalHeader);
23.     for(int i = 0; i < ntHeader.FileHeader.NumberOfSections; i++) {
24.         ReadProcessMemory(hProcess, (LPCVOID)sectAddr, &sectionHeader,
25.                           sizeof(IMAGE_SECTION_HEADER), &cbRead);
26.         if(strcmp(sectionHeader.Name, ".rsrc") == 0) {
27.             DWORD_PTR somewhere;
28.             DWORD oldProt = 0;
29.             somewhere = (DWORD_PTR)ImageBase + (DWORD_PTR)sectionHeader.VirtualAddress + 0x58;
30.             // VirtualProtectEx 将 somewhere 所在内存页设置为可执行。
31.             VirtualProtectEx(hProcess, (LPVOID)somewhere, plSize, PAGE_EXECUTE_READ, &oldProt);
32.             return somewhere;
33.         }
34.         sectAddr += sizeof(IMAGE_SECTION_HEADER);
35.     }
36.     return NULL;
37. }
```

附录：表 1 所示代码中用到的数据结构和关键函数

```
typedef struct _IMAGE_DOS_HEADER {      // DOS .EXE header
    WORD   e_magic;                     // Magic number
    WORD   e_cblp;                      // Bytes on last page of file
    WORD   e_cp;                        // Pages in file
    WORD   e_crlc;                      // Relocations
    WORD   e_cparhdr;                   // Size of header in paragraphs
    WORD   e_minalloc;                  // Minimum extra paragraphs needed
    WORD   e_maxalloc;                  // Maximum extra paragraphs needed
    WORD   e_ss;                        // Initial (relative) SS value
    WORD   e_sp;                        // Initial SP value
    WORD   e_csum;                      // Checksum
    WORD   e_ip;                        // Initial IP value
    WORD   e_cs;                        // Initial (relative) CS value
    WORD   e_lfarlc;                    // File address of relocation table
    WORD   e_ovno;                      // Overlay number
    WORD   e_res[4];                    // Reserved words
    WORD   e_oemid;                     // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                   // OEM information; e_oemid specific
    WORD   e_res2[10];                  // Reserved words
    LONG   e_lfanew;                    // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;

typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS, *PIMAGE_NT_HEADERS;

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;

typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
```

```
    DWORD   Base;
    DWORD   NumberOfFunctions; // 导出函数地址表的条目个数
    DWORD   NumberOfNames; // 导出函数名称地址表的条目个数
    DWORD   AddressOfFunctions;     // 导出函数地址表的 RVA
    DWORD   AddressOfNames;         // 导出函数名称地址表的 RVA
    DWORD   AddressOfNameOrdinals;  // 导出函数序号表的 RVA
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;

BOOL ReadProcessMemory(
    HANDLE hProcess,
    LPCVOID lpBaseAddress,
    LPVOID lpBuffer,
    DWORD nSize,
    LPDWORD lpNumberOfBytesRead)
```

参数：

- ✓ hProcess：目标进程的句柄，该句柄必须对目标进程具有 PROCESS_VM_READ 的访问权限。
- ✓ lpBaseAddress：从目标进程中读取数据的起始地址。 在读取数据前，系统将先检验该地址的数据是否可读，如果不可读，函数将调用失败。
- ✓ lpBuffer：用来接收数据的缓存区地址。
- ✓ nSize：从目标进程读取数据的字节数。
- ✓ lpNumberOfBytesRead：实际被读取数据大小的存放地址。

BeginUpdateResource/UpdateResource/EndUpdateResource 的用法说明：

- ✓ BeginUpdateResource：创建一个可执行文件的更新句柄，通过该句柄可更新该可执行文件中的资源。
- ✓ UpdateResource：使用 BeginUpdateResource 返回的句柄对可执行文件中的资源进行添加、删除和替换。
- ✓ EndUpdateResource：关闭 BeginUpdateResource 返回的句柄。在关闭后，该函数会立即将累积的资源更改写入到可执行文件中。

```
BOOL UpdateResource(
  HANDLE hUpdate,
  LPCSTR lpType,
  LPCSTR lpName,
  WORD   wLanguage,
  LPVOID lpData,
  DWORD  cb);
```

参数：

- ✓ hUpdate：BeginUpdateResource 函数返回的模块句柄，引用要更新的文件。
- ✓ lpType：要更新的资源类型。例如，RT_BITMAP 表示位图资源。
- ✓ lpName：要更新的资源的名称。此参数可以是 MAKEINTRESOURCE(ID)形式。
- ✓ wLanguage：要更新的资源的语言标识符。
- ✓ lpData：要插入到 hUpdate 指示的文件中的资源数据。
- ✓ cb：lpData 中资源数据的大小（以字节为单位）。