PE learning代码分析

姓名：袁昊男 学号：2018091618008

PE learning项目源代码主要有BaseTool.h、Control.cpp、peHeader.h以及NT.h四个源文件。其中BaseTool.h主要是病毒程序调用的工具类函数定义与实现；Control.cpp是主程序入口，用来感染“hello.exe目标程序”；peHeader.h定义了PE文件头的数据结构；NT.h为WinNT.h系统头文件。

**一、BaseTool.h**

|  |  |
| --- | --- |
| 函数名及参数 | isSuitable(char \*filename) |
| 函数功能及过程描述 | 判断目标程序是否已被感染。通过将dosheader.e\_magic、nth.Signature、infectSignature与IMAGE\_DOS\_SIGNATURE、IMAGE\_NT\_SIGNATURE、IMAGE\_INFECTED\_SINGNATURE标志位进行对应比较来判断目标程序是否为PE格式与是否被感染。 |
| 函数返回值类型 | int |
| 函数返回值及意义 | -1：打开目标文件失败；0：目标文件不是PE格式文件；1：目标文件是PE格式文件但已被感染；2：目标文件不是PE格式文件且未被感染 |

|  |  |
| --- | --- |
| 函数名及参数 | getBytes(char \* \_dst, size\_t \_len, long \_offset\_file, FILE \*fp) |
| 函数功能及过程描述 | 获取在某文件位置指定长度字节。通过调用fseek函数将文件读写指针fp定位到文件某位置\_offset\_file，并从此读取长度为\_len的字节并输出。 |
| 函数返回值类型 | int |
| 函数返回值及意义 | 获取的字节 |

|  |  |
| --- | --- |
| 函数名及参数 | setBytes(char \*\_src, int \_len, long \_offset\_file, FILE\*fp) |
| 函数功能及过程描述 | 在某文件位置写入指定长度字节。通过调用fseek函数将文件读写指针fp定位到文件某位置\_offset\_file，并从此写入长度为\_len的字节。 |
| 函数返回值类型 | int |
| 函数返回值及意义 | 写入的字节数 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_entry(FILE \*fp = NULL) |
| 函数功能及过程描述 | 获取PE头入口地址。通过调用getBytes函数：getBytes((char \*)&rst, sizeof(LONG), sizeof(IMAGE\_DOS\_HEADER)-sizeof(LONG),fp)从而返回PE头入口地址。 |
| 函数返回值类型 | long |
| 函数返回值及意义 | PE头入口地址 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_Address\_of\_EntryPoint(FILE \*fp = NULL) |
| 函数功能及过程描述 | 获取入口偏移地址。这个成员在OptionalHeader里面，OptionalHeader的类型是一个IMAGE\_OPTIONAL\_HEADER32结构。该结构总共有31个成员，占的大小为224字节。成员7就是AddressOfEntryPoint。AddressOfEntryPoint占4个字节。它表示的是代码入口的偏移地址。也就是说，把一个文件加载到内存的时候，基地址加上AddressOfEntryPoint就是代码入口地址。 |
| 函数返回值类型 | long |
| 函数返回值及意义 | 入口偏移地址 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_Number\_OF\_Section(FILE \* fp = NULL) |
| 函数功能及过程描述 | 获取文件中节的个数。NT头结构中包含有PE文件的基本信息，其中NumbgerOfSections即文件中节的个数。 |
| 函数返回值类型 | long |
| 函数返回值及意义 | 文件中节的个数。 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_IMAGE\_DOS\_HEADER(FILE \*fp = NULL) |
| 函数功能及过程描述 | 获取DOS头结构。 |
| 函数返回值类型 | IMAGE\_DOS\_HEADER |
| 函数返回值及意义 | DOS头结构。 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_IMAGE\_NT\_HEADER(FILE \* fp = NULL) |
| 函数功能及过程描述 | 获取PE文件头结构。 |
| 函数返回值类型 | IMAGE\_NT\_HEADERS |
| 函数返回值及意义 | PE文件头结构。 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_IMAGE\_SECTION\_HEADERS(\_IMAGE\_SECTION\_HEADER \*\_dst, size\_t \*\_cnt, FILE \*fp = NULL) |
| 函数功能及过程描述 | 获取PE文件节表头结构。 |
| 函数返回值类型 | void |
| 函数返回值及意义 | 无 |

|  |  |
| --- | --- |
| 函数名及参数 | get\_IMAGE\_IMPORT\_DESCRIPTORS(int \*cnt=NULL) |
| 函数功能及过程描述 | 获取函数节引入表结构。 |
| 函数返回值类型 | IMAGE\_IMPORT\_DESCRIPTOR\* |
| 函数返回值及意义 | 引入表结构地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | rva\_To\_fa(unsigned int rva) |
| 函数功能及过程描述 | 将相对虚拟地址转换为文件偏移地址。判断相对虚拟地址属于哪个节。如果RVA>=节表的RVA&&<节表内存大小，则说明属于这个节，则查看这个节在文件中的偏移地址。计算公式为：  偏移=相对虚拟地址-节的相对虚拟地址  文件偏移地址=偏移+节的偏移地址，求出的值就是相对虚拟地址在文件中的位置。 |
| 函数返回值类型 | unsigned int |
| 函数返回值及意义 | 文件偏移地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | fa\_To\_rva(unsigned int fa) |
| 函数功能及过程描述 | 将文件偏移地址转换为相对虚拟地址。判断文件偏移地址属于哪个节。如果FA>=节表的FA&&<节表内存大小，则说明属于这个节，则查看这个节在文件中的相对虚拟地址。计算公式为：  偏移=文件偏移地址-节的偏移地址  相对虚拟地址=偏移+节的相对虚拟地址，求出的值就是相对虚拟地址在文件中的位置。 |
| 函数返回值类型 | unsigned int |
| 函数返回值及意义 | 相对虚拟地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | rva\_to\_va(unsigned int rva) |
| 函数功能及过程描述 | 将相对虚拟地址转换为虚拟地址。计算公式为：  虚拟地址=基地址+相对虚拟地址。 |
| 函数返回值类型 | unsigned int |
| 函数返回值及意义 | 虚拟地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | va\_to\_rva(unsigned int va) |
| 函数功能及过程描述 | 将虚拟地址转换为相对虚拟地址。计算公式为：  相对虚拟地址=虚拟地址-基地址。 |
| 函数返回值类型 | unsigned int |
| 函数返回值及意义 | 相对虚拟地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | getKernal32Base() |
| 函数功能及过程描述 | 获取Kernel32.dll的基地址。采用通过TEB获得PEB结构地址，然后再获得PEB\_LDR\_DATA结构地址，然后遍历模块列表，查找kernel32.dll模块的基地址。参考：  https://blog.csdn.net/whypp/article/details/5681172中第三种方法。 |
| 函数返回值类型 | dword |
| 函数返回值及意义 | Kernel32.dll的基地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | getAddressOfGetProcAddre() |
| 函数功能及过程描述 | 获取函数API入口地址。对于给定的API，搜索其地址可以直接通过Kernel32.dll的引出表信息搜索，同样也可以先搜索出GetProcAddress和LoadLibrary两个API函数的地址，然后利用这两个API函数得到所需要的API函数地址。 |
| 函数返回值类型 | dword |
| 函数返回值及意义 | 函数API的入口地址。 |

|  |  |
| --- | --- |
| 函数名及参数 | infect(FILE \* fp=NULL) |
| 函数功能及过程描述 | 感染目标程序。首先判断是否为PE格式文件。如果是未感染的PE格式文件，则调用finalOfSections\_fva()函数将指针定位到节表最后一个位置的文件偏移处，获得文件偏移地址，将其转化为相对虚拟地址，写入程序的入口点AddressOfEntry。然后将病毒节写入节表，更新节表中相关参数的信息，最后添加感染标记，将当前位置设为文件结尾。 |
| 函数返回值类型 | int |
| 函数返回值及意义 | -1：打开目标文件失败；0：目标文件不是PE格式文件；1：目标文件是PE格式文件但已被感染；2：目标文件不是PE格式文件且未被感染。 |

|  |  |
| --- | --- |
| 函数名及参数 | uninfect() |
| 函数功能及过程描述 | 将程序标记为“未感染”。获取目标程序的DOS头入口地址，将infectedSign是否感染标志位写位0。 |
| 函数返回值类型 | void |
| 函数返回值及意义 | 无 |

**二、Control.cpp**

|  |  |
| --- | --- |
| 函数名及参数 | main() |
| 函数功能及过程描述 | 感染程序主程序。首先打开待感染文件“hello.exe”。根据flag标志判断文件是否被感染。若flag为2，则调用infect()函数执行感染程序；若flag为1，则调用uninfect()函数，将文件标记为“未感染”。 |
| 函数返回值类型 | int |
| 函数返回值及意义 | 0，无意义。 |

**三、感染步骤**

1. 判断感染标记，若目标文件未被感染，则继续执行感染程序；否则退出。
2. 通过比较dosh.e\_magic与0x5a4d判断目标文件开始两个字节是否为“MZ”。目标文件不是MZ格式文件则退出感染函数。
3. 通过比较fileh.Signature与0x4550判断目标文件是否为PE格式文件。目标文件不是PE格式文件则退出感染程序。
4. 获得Directory（数据目录）的个数，每个数据目录信息占8个字节。
5. 计算节表起始位置：节表起始地址=Directory的偏移地址+数据目录占用的字节数。
6. 计算当前节表末尾的偏移量（紧接其后用于写入一个新的病毒节）：当前节表末尾偏移量=节表起始位置+节的个数×（每个节表占用的字节数28H）。
7. 更新节表：
8. 写入新节名（8字节）；
9. 写入新节的实际字节数（4字节）；
10. 写入新节在内存中的偏移地址（4字节），同时计算病毒节的入口地址；
11. 写入新节在文件中对齐后的大小；
12. 写入新节在文件中的开始地址：开始地址=上节在文件中的开始位置+上节对齐后的大小；
13. 修改映像文件头中的节表数。
14. 修改AddressOfEntryPoint（即将程序入口点修改为指向病毒入口位置），同时保存旧的AddressOfEntryPoint，以便返回原程序继续执行。
15. 更新SizeOfImage：内存中整个PE文件映像大小=原SizeOfImage+病毒节经过内存节对齐后的大小。
16. 写入感染标记。
17. 写入病毒代码并将其添加到新节中：ECX=病毒代码长度，ESI=病毒代码入口地址，EDI=病毒节写入地址。
18. 将当前文件位置设为文件末尾。