Polytechnic UNIVERSITY

Scholarship For Service

PE File Infection Techniques
Part 1

Konstantin Rozinov
(krozinov@yahoo.com)

Polytechnic University
Scholarship for Service

Overview

- Introduction
- What is Win32?
- The PE File Format
- Windows Viruses
  - *Overwriting, Header Infectors, prepending, appending, EPOs, DLL infectors, companion infection, cavity infections, etc.*
- Closing Comments
- References

Introduction

- Most 32-bit Windows viruses infect executables and object files (DLLs).
- Most often they achieve this by modifying the PE (Portable Executable) file format of the executable.
- Virus writers have become very sophisticated.

What is Win32?

- Win32 refers to the application programming interface (API) available in Windows 95, 98, NT, and 2000.
- It is the set of system functions that are part of the operating system and that are available to be called from a 32-bit Windows application.
- Although the functions may be implemented differently on different Windows platforms, it is possible to write one program that will run on all Windows platforms that support Win32.
  - This fact sometimes allows viruses to spread quicker.

The PE File Format

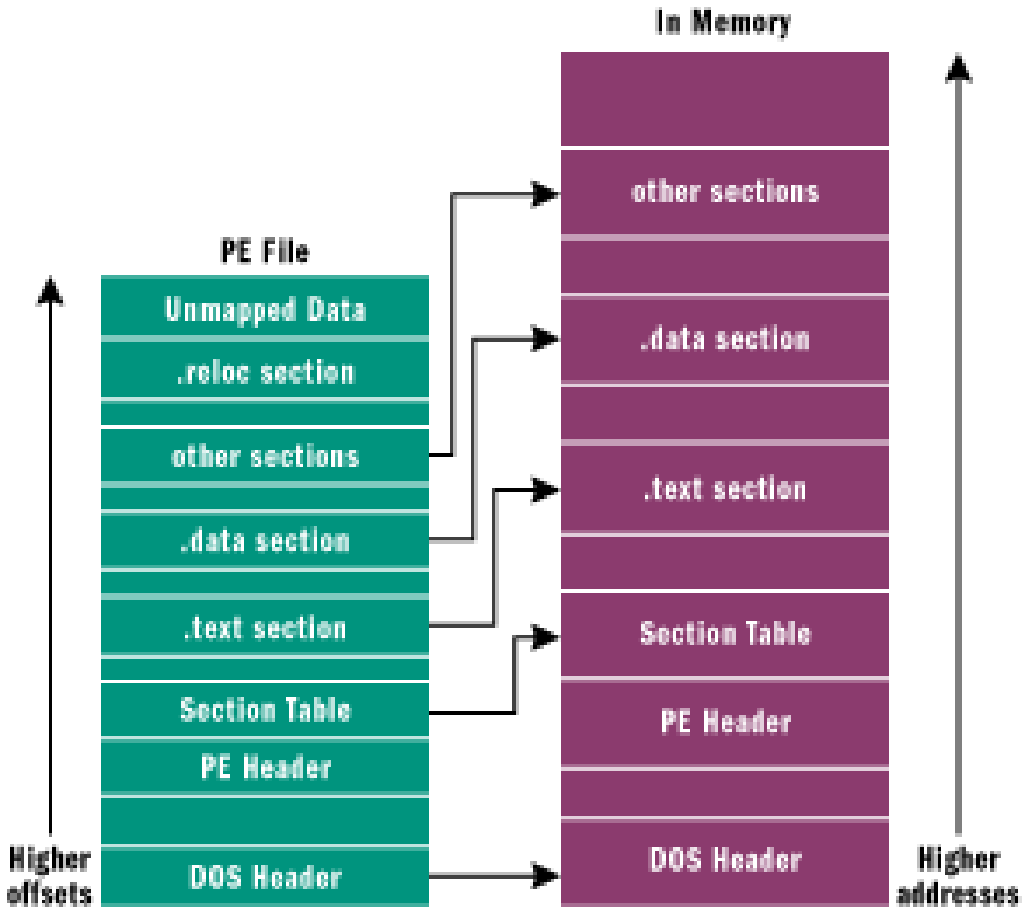
- What is Portable Executable (PE) file format?
  - It is Microsoft's format for 32-bit executables and object files (DLLs).
    - based on COFF (Common Object File Format).
    - Compatible across all 32-bit Windows operating systems.
  - Like other formats, PE has different areas called sections, which contain code or data.
    - Typical sections include: `.text`, `.data`, `.rdata`, `.bss`, & `.reloc`.
    - It has a PE header which is vital in detecting viruses.

The PE File Format *(cont'd)*

- PE structure:
  - PE files loaded into memory are almost identical to how they look on disk.
  - Easy to find sections:
    - Base Address (typically `0x00400000`).
    - RVA (Relative Virtual Address) is an offset from the BA.
  - Has a DOS stub program which is executed only when the executable is run in MS-DOS.
    - Prints out the message "This program cannot be run in DOS mode."
    - At location `0x3c` (60th byte), the stub has the file offset to the Portable Executable (PE) signature ("`PE\0\0`").

The PE File Format *(cont'd)*

- PE structure:



The PE File Format *(cont'd)*

- Important fields in the PE Header:
  - `Machine`
    - Which CPU this file intended for (almost always `0x14c`).
    - Checked by viruses to ensure they only infect x86 platforms.
  - `NumberOfSections`
    - The number of sections in the file.
    - If virus adds a new section, this field is updated, as is the Section Table.
  - `Characteristics`
    - What type of file this is (EXE vs. DLL).
  - `SizeOfCode`
    - Size of all the code sections.
    - Some viruses neglect to change this field if new code is added.

The PE File Format *(cont'd)*

- Important fields in the PE Header:
  - `AddressOfEntryPoint`
    - The relative virtual address (RVA) where execution begins (`.text`).
    - Many viruses change this to point to the virus code.
  - `ImageBase`
    - First byte of image in memory (typically `0x00400000`).
    - Used by viruses to calculate various addresses.
  - `SizeOfImage`
    - The size of the image (originally calculated by the linker).
    - Most viruses update this value, as its checked by the NT loader.
  - `Checksum`
    - Sometimes used by viruses to avoid double-infections.

The PE File Format *(cont'd)*

- The Section Table:
  - Contains information about each individual section in what is known as section headers including:
    - `VirtualSize`: Total size of the section in memory.
    - `SizeOfRawData`: Size of the section on disk.
    - `Characteristic`: What kind of section this is (i.e. code or data). Typically, sections that are executable are not writeable, but virus sections need to be both executable and writeable because its data is inside the same section.
  - Viruses either add their own headers or update existing headers in order to add their malicious code to the executable.

The PE File Format *(cont'd)*

- The Imports Table:
  - When you use code or data from another DLL, you're importing it.
  - The Windows loader locates all of the imported functions and data and makes those addresses available to the executable (via `.idata` section).
  - Virtually all executables have an `.idata` section.
  - Viruses use the import table to lookup the address of any API function they need to call.
    - This is done via `GetProcAddress` calls.

Windows Viruses

- Overwriting:
  - Simplest type of virus.
  - Replaces code of host program, rendering it useless and unrecoverable, with the virus code.
  - Easily detectable since programs will stop functioning.

Windows Viruses *(cont'd)*

- Header Infection:
  - The virus inserts itself between the PE header and the beginning of the first section.
  - Modifies `AddressOfEntryPoint` field in PE header to point to the virus code instead.
  - These types of viruses have to be very small.
    - Sections have to start at an offset which is a multiple of the `FileAlignment`.
    - `FileAlignment` has default value of 512 bytes, but can be as large as 64 kilobytes.
  - Example: Win95/Murky.

Windows Viruses *(cont'd)*

- Prepending:
  - These types are usually written in a HLL (like C).
  - Two ways of infecting:
    1. Takes the PE header and moves it to the end of the host program. Then inserts virus code into the beginning of host program.

    - or -

    2. The virus appends the host program to itself.
  - In either case, the original host program is executed after the virus.
    - May use a clean copy of host program.
    - May clean infected file, launch it, and then re-infect it.

Windows Viruses *(cont'd)*

- Appending without adding new Section Header:
  - Does not add a new section header to the section table, but instead patches the last section header in the section table:
    - Modifies `SizeOfImage`, `VirtualSize` and `SizeOfRawData` to include size of virus code and reflect the new size of file.
    - Modifies `AddressOfEntryPoint` field to point to the virus body and changes the `Characteristic` of the last section to be executable.
    - `NumberOfSections` field not changed.
  - Example: Win95/Anxiety.

Windows Viruses *(cont'd)*

- Appending without modifying `AddressOfEntryPoint` (Entry Point Obfuscation or EPO):
  - Calculates where the original `AddressOfEntryPoint` points to and places a `JMP` instruction there which will point to the virus code.
    - Harder to write such a virus because of relocation issues.
      - Virus has to take care of relocation entries which are pointing into overwritten part of host file.
    - `JMP` instruction is not necessarily the first instruction.
      - Can be preceded by garbage code.
  - Example: Win32/Cabanas and Win95/Marburg.

Windows Viruses *(cont'd)*

- Companion:
  - Do not modify the host file.
  - Creates a copy of the virus with the same name as the host file.
    - `.com` extension is used for the virus copy.
    - Windows always executes `.com` files before `.exe` files.
    - Makes use of the following Win32 API calls:
      - `FindFirstFileA`, `FindNextFileA`, `CopyFileA`, `CreateProcessA`
    - Executes the virus code, followed by the original host file.
  - Example: Win95/Spawn.4096.

Windows Viruses *(cont'd)*

- Cavity:
  - All PE files have slack space in them.
    - Slack space exists between sections and it's usually filled with zeroes.
      - Thus the difference between `VirtualSize` (smaller) and `SizeOfRawData` (larger).
    - The slack space is there because all sections have to start at the alignment specified by the `FileAlignment` field.
      - Default value of `FileAlignment` is 512 bytes.
      - Although this is quite small, there is usually multiple slack areas, adding up to a larger chunk of space.
  - Example: Win95/CIH.

Windows Viruses *(cont'd)*

- Cavity *(cont'd)*:
  - The virus is split up into several pieces and placed into several slack areas between sections.
  - The loader code for the virus, which "rebuilds" the virus, can be very small.
  - The virus also changes the `VirtualSize` of the section in the Section Table to be the same as the raw size.
    - The size of the host file does not get bigger, since the `SizeOfRawData` has not been changed.
    - Harder to detect, but also harder to write.

Windows Viruses *(cont'd)*

- DLL Infectors:
  - Do not attack the `AddressOfEntryPoint` because DLLs are normally started at the specified `DLLEntry` point.
    - For example calling `WriteFile` from `KERNEL32.DLL` will start from a specific address inside the DLL.
  - So instead, the virus will patch the RVA of an API function (like `WriteFile`) in the DLL's Exports Table to point to the virus code.
    - Checksum is usually recalculated and patched back into the DLL since the NT loader checks that.
  - Example: Win95/Lorez.

Closing Comments

- There are other ways of infecting PE files.
  - For example, hooking the Imports Table of the host program (via `.idata` section).
  - Appending to multiple sections at the same time.
  - Shifting sections and then inserting the virus into the resulting hole.
  - Replacing certain `CALL`s or `PUSH`es with `JMP`s to the virus code.
- There are also encrypted viruses, packed viruses, & polymorphic viruses.

References

- Szor, Peter. Attacks on Win32. Virus Bulletin Conference, October 1998, Munich/Germany, page 57-84.

- Rozinov, Konstantin. Reverse Engineering: An In-Depth Analysis of the Bagle Virus. August 2004: http://rozinov.sfs.poly.edu/papers/bagle_analysis_v.1.0.pdf.

- Classic Viruses from Viruslist.com: http://www.viruslist.com/en/viruses/encyclopedia?chapter=152540474

References *(cont'd)*

- Frequently Asked Questions about Win32 Programming: http://www.iseran.com/Win32/FAQ/faq.htm.

- Inside Windows: An In-Depth Look into the Win32 Portable Executable File Format: http://msdn.microsoft.com/msdnmag/issues/02/02/PE/default.aspx.

- Microsoft Portable Executable and Common Object File Format Specification: http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx.