

EECS 444 Homework 2: Mimic You - Malware!

Yida Liu

October 12, 2019

1 Write a program using c to implement the function

See ./PE-Import.c for the source of the program.

We used the gcc compiler included in mingw-x86_64 to compile the program to ./PE-Import.exe.

2 Check the Import Table of ./PE-Import.exe

We used Exeinfo PE to check the import table. Here is a screenshot output of the Import Table.

The screenshot displays the Exeinfo PE tool interface. The left pane shows the 'Imports' section with a table of imported DLLs and functions. The right pane shows the 'File Information' section with various metadata.

Imports:

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
KERNEL32.dll	0000703C	00000000	00000000	000075A4	00007124
msvcrt.dll	0000709C	00000000	00000000	00007638	00007184

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
000070AC	00002EAC	0000741A	0000	__p__fmode
000070B0	00002EB0	00007428	0000	__set_app_type
000070B4	00002EB4	0000743A	0000	__setusermatherr
000070B8	00002EB8	0000744E	0000	__access
000070BC	00002EB8	00007458	0000	__amsg_exit
000070C0	00002EC0	00007466	0000	__cexit
000070C4	00002EC4	00007470	0000	__initterm
000070C8	00002EC8	0000747C	0000	__job
000070CC	00002ECC	00007484	0000	__onexit
000070D0	00002ED0	0000748E	0000	abort
000070D4	00002ED4	00007496	0000	calloc
000070D8	00002ED8	000074A0	0000	exit
000070DC	00002EDC	000074A8	0000	fclose
000070E0	00002EE0	000074B2	0000	fopen
000070E4	00002EE4	000074BA	0000	fprintf
000070E8	00002EE8	000074C4	0000	fputs
000070EC	00002EE8	000074C4	0000	fputs
000070F0	00002EEC	000074CC	0000	free
000070F4	00002EF0	000074D4	0000	free
000070F8	00002EF4	000074DC	0000	fseek
000070FC	00002EF8	000074E4	0000	ftell
00007100	00002EFC	000074EC	0000	fwrite
00007104	00002F00	000074F6	0000	malloc
00007108	00002F04	00007500	0000	memcpy
0000710C	00002F08	0000750A	0000	printf
00007110	00002F0C	00007514	0000	signal
00007114	00002F10	0000751E	0000	strlen
00007118	00002F14	00007528	0000	strncmp
0000711C	00002F18	00007532	0000	strstr
00007120	00002F1C	0000753C	0000	vfprintf

File Information:

File: PE-Import.exe
Entry Point: 000014A0
File Offset: 000008A0
Linker Info: 2,30
File Size: 0000C6CDh
Image is 32bit executable
RES/OVL: 0 / 48 %
2019
GCC MINGW-64w compiler for 32 / 64 bit Windows (exe) - http://ming
Lamer Info - Help Hint - Unpack info
Not packed, try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-ray

File Information (Right Pane):

From header: Very often:
Size of headers: 00000400 400 or 1000
Size of optional header: 00E0 00E0
Number of Dirs: 0010 0010h
Base of Code: 00001000 00001000
Image Base: 00400000 00400000
Magic optional header: 010B 010B 32bit
Debugger Info - size: No
File offset to PE: 0080 click me
Checksum CRC: 0001398A 00000000
Machine type: 0x14C Intel I386 (same ID used for 4
OS version: 4.0 4.0 Win NT 4.0
From file: 4.0
Image version: 1.00
File / sec-n alignment: 0200 / 1000
Entry Point to End of File bytes: 23904 = 23.34 KB

3 Original vs. Packed

3.1 Use UPX to pack PE file

We used the following command using UPX to pack the compiled program to PE-Import-upx.exe.

```
# Pack using UPX
```

```
upx -9 PE-Import.exe -o PE-Import-upx.exe
```

3.2 Check the Import Table of the Packed Executable

Here is a screenshot output of the Import Table of ./PE-Import-upx.exe

The screenshot displays two windows. The left window shows the 'Imports' section of a PE file, listing imported DLLs and their addresses. The right window shows the 'Exeinfo PE' tool interface, which provides detailed information about the packed executable, including file size, entry point, and various headers.

Imports:

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
KERNEL32.DLL	00000000	00000000	00000000	00013058	0001303C
msvrt.dll	00000000	00000000	00000000	00013065	00013050

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
00013050	00002E50	000130AC	0000	_job

Exeinfo PE - ver.0.0.4.8 by A.S.L - 999+64 sign 2017.09.08

File: PE-Import-upx.exe
Entry Point: 00012870 EP Section: UPX1
File Offset: 00002A70 First Bytes: 60.BE.15.00.41
Linker Info: 2.30 SubSystem: Win Console
File Size: 000090CDh Overlay: 000060CD
Image is 32bit executable RES/OVL: 0 / 66 % 2019
UPX -> Markus & Laszlo ver. [3.95] <- from file. (sign like UPX packer)
Lamer Info - Help Hint - Unpack info
unpack "upx.exe -d" from http://upx.github.io or any UPX/Generic un

Code: 003000h - decimal: 12 KB

From header: Very often:
Size of headers: 00001000 400 or 1000
Size of optional header: 00E0 00E0
Number of Dirs: 0010 0010h
Base of Code: 00010000 00001000
Image Base: 00400000 00400000
Magic optional header: 010B 010B 32bit

Debugger Info - size: No
File offset to PE: 0080 click me
Checksum CRC: 00000000 00000000

Machine type: 0x14C Intel I386 (same ID used for 4
OS version: 4.0 4.0 Win NT 4.0

From file:
Image version: 1.00 4.0
File / sec-n alignment: 0200 / 1000
Entry Point to End of File bytes:
1424 = 1.39 KB

File icon:

3.3 Use UPX to unpack the Packed Executable

We used the following command to use UPX to unpack the packed executable to PE-Import-upx-unpacked.exe

```
# Unpack using UPX
```

```
upx -d PE-Import-upx.exe -o PE-Import-upx-unpacked.exe
```

Here is a screenshot output of the Import Table of ./PE-Import-upx-unpacked.exe

Imports :

DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
KERNEL32.DLL	00000000	00000000	00000000	000075A4	00007124
msvrt.dll	00000000	00000000	00000000	00007638	00007184

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name
00007184	00002F84	000073E0	0000	__getmainargs
00007188	00002F88	000073F0	0000	__initenv
0000718C	00002F8C	000073FC	0000	__conv_init
00007190	00002F90	0000740C	0000	__p_acmdln
00007194	00002F94	0000741A	0000	__p_fmode
00007198	00002F98	00007428	0000	__set_app_type
0000719C	00002F9C	0000743A	0000	__setusermatherr
000071A0	00002FA0	0000744E	0000	__access
000071A4	00002FA4	00007458	0000	__amsg_exit
000071A8	00002FA8	00007466	0000	__cexit
000071AC	00002FAC	00007470	0000	__initterm
000071B0	00002FB0	0000747C	0000	__job
000071B4	00002FB4	00007484	0000	__onexit
000071B8	00002FB8	0000748E	0000	abort
000071BC	00002FBC	00007496	0000	calloc
000071C0	00002FC0	000074A0	0000	exit
000071C4	00002FC4	000074A8	0000	fclose
000071C8	00002FC8	000074B2	0000	fopen
000071CC	00002FCC	000074BA	0000	fprintf
000071D0	00002FD0	000074C4	0000	fputs
000071D4	00002FD4	000074CC	0000	fread
000071D8	00002FD8	000074D4	0000	free
000071DC	00002FDC	000074DC	0000	fseek
000071E0	00002FE0	000074E4	0000	ftell
000071E4	00002FE4	000074EC	0000	fwrite
000071E8	00002FE8	000074F6	0000	malloc
000071EC	00002FEC	00007500	0000	memcpy
000071F0	00002FF0	0000750A	0000	printf
000071F4	00002FF4	00007514	0000	signal
000071F8	00002FF8	0000751F	0000	strlen

Exeinfo PE - ver.0.0.4.8 by A.S.L - 999+64 sign 2017.09.08

File : PE-Import-upx-unpacked.exe

Entry Point : 000014A0 EP Section : .text

File Offset : 000008A0 First Bytes : 83.EC.0C.C7.0!

Linker Info : 2.30 SubSystem : Win Console

File Size : 0000C6CDh Overlay : 000060CD

Image is 32bit executable RES/OVL : 0 / 48 % 2019

GCC MINGW-64w compiler for 32 / 64 bit Windows (exe) - http://mingw.org

Lamer Info - Help Hint - Unpack info

Not packed, try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-ray

Size of Code : 001A00h - decimal : 6.5 KB

Not used 1970-01-01

(06) .idata 1970-01-01

0 % of exe Nr of ID : 0

not Signed

PE 1970-01-01

(03) .rdata

Not used

Not used

.NET Meta Directory

From header : Very often :

Size of headers : 00001000 400 or 1000

Size of optional header : 00E0 00E0

Number of Dirs : 0010 0010h

Base of Code : 00001000 00001000

Image Base : 00400000 00400000

Magic optional header : 010B 010B 32bit

Debugger Info - size : No

File offset to PE : 0080 click me

Checksum CRC : 00000000 00000000

Machine type : 0x14C Intel I386 (same ID used for 4)

OS version : 4.0 4.0 Win NT 4.0

From file

Image version : 1.00

File / sec-n alignment : 0200 / 1000

Entry Point to End of File bytes : 23904 = 23.34 KB

File icon :

4 Fool Anti-Malware Scanner

First, we used the Virustotal to scan the original PE-Import.exe. Virustotal reports that 1 out of 70 engines reported unsafe of this program.

Our technique is quite simple and straight forward. We use the previous ./PE-Import-upx.exe and use PElOCK to pack an additional layer of protection in ./PE-Import-upx-pelock.exe . Our justification is that virus / trojan often employes multiple layers of packing to hide the actual code underneath it. This file was detected malicious and dangerous by 28 out of 70 engines in VirusTotal.