EECS 444 Homework 2: Mimic You - Malware!

Yida Liu

October 12, 2019

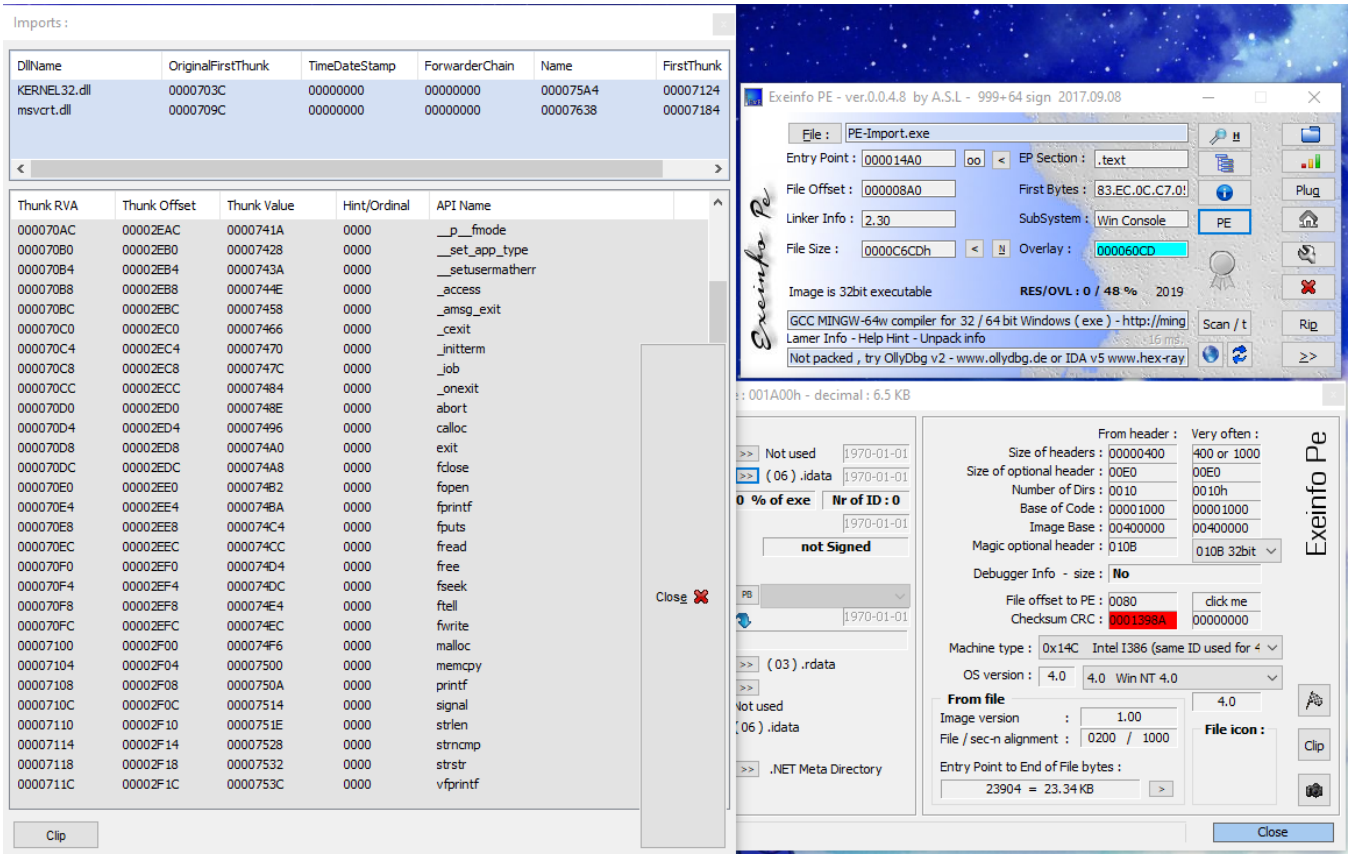
1 Write a program using c to implement the function

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.



3 Original vs. Packed

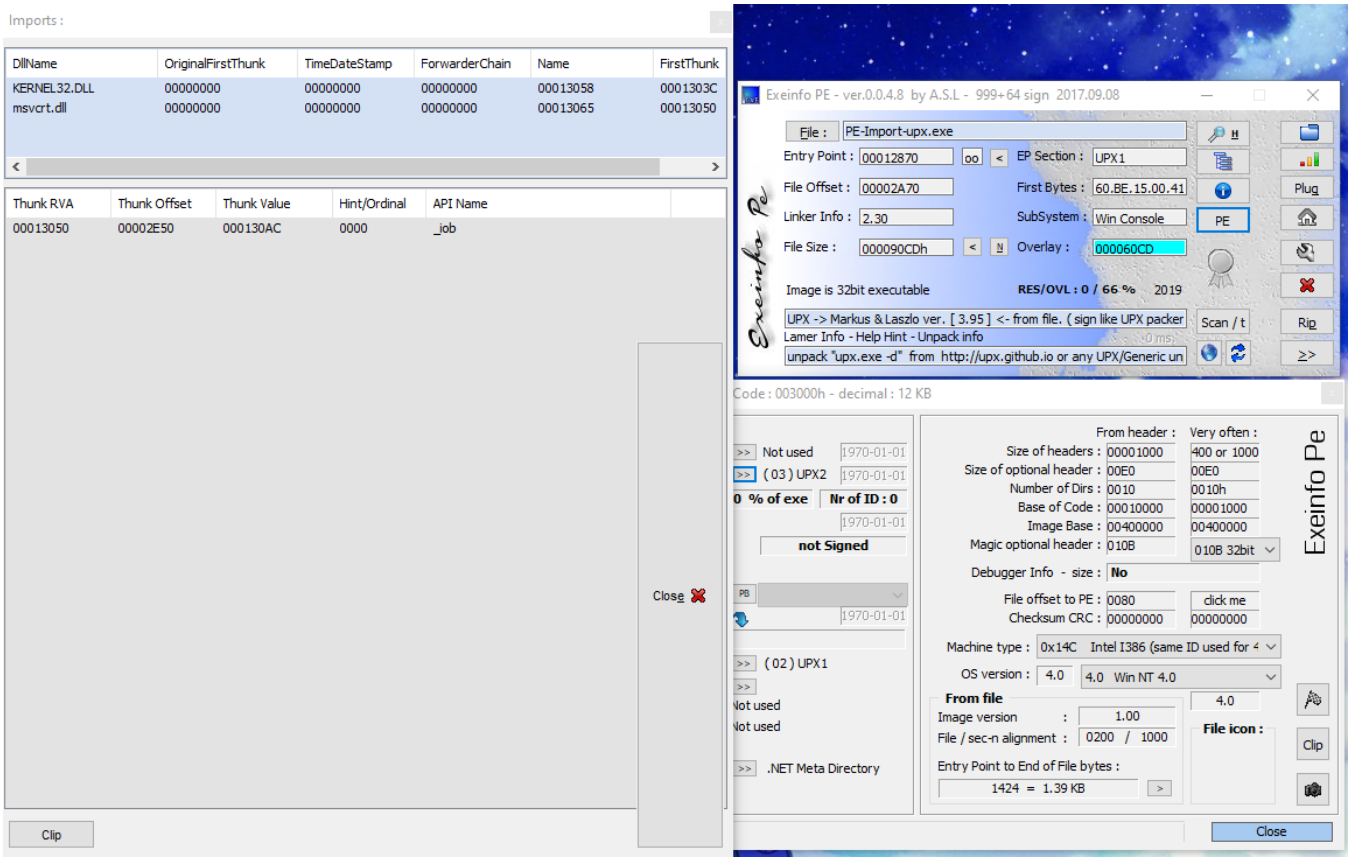
3.1 Use UPX to pack PE file

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`

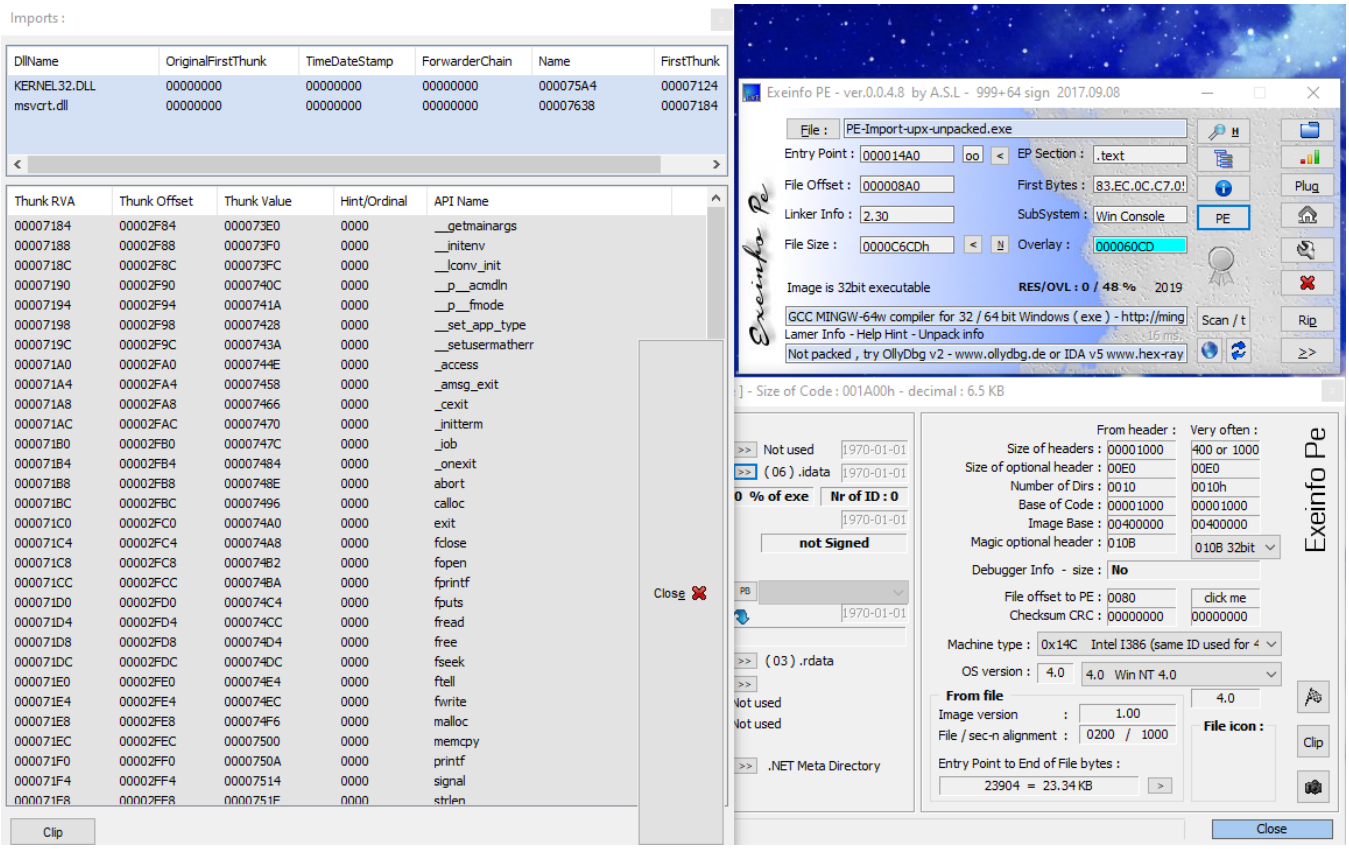


3.3 Use UPX to unpack the Packed Executable

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`



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 used Stunnix cxx-obfus tools to obfuscate the source code and compile to executable.

```
# Use Stunnix to obfuscate the program
cxx-obfus PE-Import.c -o PE-Import.obfs.c -x xpg4
# Compile to exe
gcc PE-Import.obfs.c -o PE-Import.obfs.exe
```

The executable application passed the virus check of VirusTotal.

4.1 Additional Ideas

We also imagined additional ways / ideas to hide the malicious identity of our application.

1. Self-Interpreting

   Our program could be factored into an interpreter with specialized instruction set / opcodes and a program written in the specific interpreter language. Without elaboration, it would be hard to reverse-engineered. In fact, tigress implements such an idea to obfuscate C program.

2. Intermediate Representation

We could convert the c program to LLVM Intermediate Representation(IR), an platform-independent low-level assembly language and perform additional obfuscation from there, after which the obfuscated IR code could be compiled to machine code at any supported hardware platforms.