

CSC 471 Spring 2025 Lab 3

Dr. Si Chen

Build a heuristic malware detection system

The goals of this lab:

- Understanding the concepts of IAT/EAT
- Know how to use static analysis to detect malware

Objectives and Targets

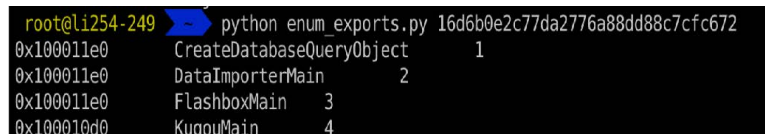
Please download **malware.zip** from our class website, unzip it. It should release the following malware samples:

- 16d6b0e2c77da2776a88dd88c7cfc672
- 0fd6e3fb1cd5ec397ff3cdbaac39d80c
- 6a764e4e6db461781d080034aab85aff
- cc3c6c77e118a83ca0513c25c208832c
- e0bed0b33e7b6183f654f0944b607618
- 1c1131112db91382b9d8b46115045097

Please **create a runnable program** (recommend using Python).

This program should be able to scan a folder, and analysis the PE structure of each malware sample.

Then, implement the following heuristic rules:



```
root@li254-249 ~# python enum_exports.py 16d6b0e2c77da2776a88dd88c7cfc672
0x100011e0 CreateDatabaseQueryObject 1
0x100011e0 DataImporterMain 2
0x100011e0 FlashboxMain 3
0x100010d0 KugouMain 4
```

Figure 1: Malware Sample 1 Export Functions Information

1. If three or more export functions have the same memory address, it's a malware.

E.g. In Figure.1, all three export function have the same memory address 0x100011e0 (*CreateDatabaseQueryObject*, *DataImporterMain*, *FlashboxMain*), so it's a malware.

```

root@li254-249:~# python enum_exports.py cc3c6c77e118a83ca0513c25c208832c
0x10001100 LpkPresent 1
0x10001120 ScriptApplyDigitSubstitution 2
0x10001140 ScriptApplyLogicalWidth 3
0x10001160 ScriptBreak 4
0x10001180 ScriptCpToX 5
0x100011a0 ScriptCacheGetHeight 6
0x100011c0 ScriptFreeCache 7
0x100011e0 ScriptGetCMap 8
0x10001200 ScriptGetFontProperties 9
0x10001220 ScriptGetGlyphABCWidth 10
0x10001240 ScriptGetLogicalWidths 11
0x10001260 ScriptGetProperties 12
0x10001280 ScriptIsComplex 13
0x100012a0 ScriptItemize 14
0x100012c0 ScriptJustify 15
0x100012e0 ScriptLayout 16
0x10001300 ScriptPlace 17
0x10001320 ScriptRecordDigitSubstitution 18
0x10001340 ScriptShape 19
0x10001360 ScriptStringAnalyse 20
0x10001380 ScriptStringCpToX 21
0x100013a0 ScriptStringFree 22
0x100013c0 ScriptStringGetLogicalWidths 23
0x100013e0 ScriptStringGetOrder 24
0x10001400 ScriptStringOut 25
0x10001420 ScriptStringValidate 26
0x10001440 ScriptStringXtoCP 27
0x10001460 ScriptString_pLogAttr 28
0x10001480 ScriptString_pSize 29
0x100014a0 ScriptString_pcOutChars 30
0x100014c0 ScriptTextOut 31
0x100014e0 ScriptXtoCP 32
0x10001890 ServiceMain 36
0x10001500 UspAllocCache 33
0x10001520 UspAllocTemp 34
0x10001540 UspFreeMem 35

```

Figure 2: Malware Sample 2 Export Functions Information

2. If three or more export functions have the same memory offset (the difference between two export functions are the same), it's a malware.

E.g. In Figure.2, the memory offset (difference) between each export functions is always 0x20, so it's a malware.

When running your program, it should be able to scan through all malware samples, and output which rules that malware sample violate.

Experiment Setup

1. Check the tutorial on website to create an account and log in Badger CTF system (via SSH)
2. Download the malware.zip (password: infected) from our course website (the following command should be typed in one line).

```
wget --no-check-certificate
https://www.cs.wcupa.edu/schen/malware25/lab3/malware.zip
```

3. Download Python script: enum_export.py and start from there. You can re-use this script in your project. P.S. You can type the following command inside Badger CTF to download the script (the following command should be typed in one line).

```
wget --no-check-certificate  
https://www.cs.wcupa.edu/schen/malware25/download/enum\_export.py
```

4. Unzip the malware.zip and make sure the malware sample and the Python script enum_export.py are in the same folder.

```
unzip malware.zip
```

5. Type the following command in your terminal, and hit enter key, it should output the export functions and addresses.

```
python3 enum_export.py 16d6b0e2c77da2776a88dd88c7cfc672
```

6. Now tweak the Python script (enum_export.py), read the code and figure out its meaning.
7. Write Python code (You can use Vim editor) to implement all the functionalities.
8. You can check the online documentation of the *pefile* library for more details

```
https://github.com/erocarrera/pefile/blob/wiki/UsageExamples.md#introduction
```

9. You can use AI to help you coding or writing. However, you should explicitly indicate which portions of the code/report were generated by the AI tool at the end of your report.

Submission

- The lab due date is available on our course website. Late submission will not be accepted;
- The assignment should be submitted to D2L directly.
- Your submission should include: A **detailed project report in PDF format** to describe what you have done, including screenshots of the final result
- **No copy or cheating is tolerated.** If your work is based on others', please give clear attribution. Otherwise, you **WILL FAIL** this course.