CS4238 Assignment 3: Static and Dynamic Analysis of Malware

1. Instructions

Due date & time:

Wednesday, 21st April 2024, 23:59

SGT. **Instructions:**

- This is an **individual** assignment. You MUST finish the implementation and report **independently**.
- Malware samples are in `A3-files.zip`. The zip file's password: `infectednus`
- Make sure you analyze these samples using your malware analysis tools ***only inside*** **a safe environment** as discussed in the class!
- Submission:
  - Submission has to be made as a single zip file to Canvas.
  - Prepare a word/PDF document for your report that answers the questions below concisely.
  - For Task 4, also include the Python code as instructed below.
- There will be **no deadline extensions**, and there will be **penalties** for late submissions as follows:
  - Late up to 6 hours: You will be evaluated for a maximum of 90%.
  - Later than 6 hours but no later than 1 day: You will be evaluated for a maximum of 80%.
  - Later than 1 day but no later than 2 days: You will be evaluated for a maximum of 70%.
  - Later than 2 days (*subject to approval*): You will be evaluated for a maximum of 60%.

2. Assignment Tasks

Basic Static Analysis (10 marks)

Recommended Reading:

Chapters 0 and 1 from the "Practical Malware Analysis" textbook.

Task 1 (5 marks, 1 mark for each question): Answer the following questions by analyzing `HW-A-1.exe` using basic static analysis techniques only.

1. Upload the programs to https://www.virustotal.com and check if they match any existing antivirus definition?
2. Are there any indications that these files are packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it.
3. When was the program compiled?
4. Do any of the imports hint at the program's functionality? If so, which imports are they, and what do they tell you?
5. What host- or network-based indicators can you use to identify these malwares on infected machines?

Task 2 (5 marks, 1 mark for each question): Answer the following questions by analyzing `HW-A-2.exe` using basic static analysis techniques only.

1. Upload the programs to https://www.virustotal.com and check if they match any existing antivirus definition?
2. When was the program compiled?
3. Do any of the imports hint at the program's functionality? If so, which imports are they, and what do they tell you?
4. What host- or network-based indicators can you use to identify the malware on infected machines?
5. The file has multiple resources in its resource section. What are their respective MD5 or SHA hashes? What are the differences between the resources? [Hint: Resources are usually in BIN format]

Basic Static and Dynamic Analysis (5 marks)

Recommended Reading:

Chapters 2 and 3 from the "Practical Malware Analysis" textbook.

Task 3 (5 marks, 1 mark for each question): Answer the following questions by analyzing `HW-A-3.exe` using basic static and dynamic analysis techniques only.

1. What is this program's functionality and explain the basis for this guess?
2. What are your observations about the program using Basic Static Analysis techniques?
3. What are your observations about the program through dynamic analysis?
4. List the potential host-based of this malware.
5. List the potential network-based indicators of this malware? To which domains does the malware possibly connect?

PE File Format (5 marks)

PEfile Usage Examples: https://github.com/erocarrera/pefile/blob/wiki/UsageExamples.md

Task 4 (5 marks, 1 mark for each question): Write a Python program that uses the `pefile` API (https://code.google.com/p/pefile/). The program takes a PE file as input from the command line, and should perform the operations below. We have provided a template Python program (`A3.py`) that you can use to get started. **Note that the template is provided as a reference and you should not rely on its implementation correctness, although we have ensured this to some extent.** You may modify or completely rewrite the template if you wish. (**Note**: In addition to answering the questions below, please attach screenshots of the results in your report. Additionally, include your code in a single Python file inside your submitted zip file.)

1. Write a program to output the following to standard output:
   a. Identify the file type as `DLL`, `EXE`, or `SYS` regardless of the file's extension. [Answer to this is provided in `A3.py` for you to get started]
   b. The total number of imported `DLLs`.
   c. The total number of imported functions.

    d. The compile time.

2. Alert the user if the code's entry point is not in a section with the name `.text`, `.code`, `CODE`, or `INIT`. (**Hint**: Aforementioned usage examples may have some relevant code snippets that you can use. You can consider using `pe.OPTIONAL_HEADER.AddressOfEntryPoint` to get the address of the entry point. You can use `section.contains_rva()` for your checking.)
3. Use the PEiD database that comes with `pefile` to identify packers. Confirm that this works with UPX. Output the detection to standard output.
4. Calculate and output the entropy for each section. Alert the user if there is a suspicion that a section may be packed or compressed (if the section's entropy >=6).
5. Compare the PE Optional Header checksum with the actual checksum. Alert the user when they do not match up.