Malware Static Analyzer Final Project Report

Savannah Alfaro
Dept. of Computing Security
Rochester Institute of Technology
Email: sea2985@rit.edu

I. Introduction

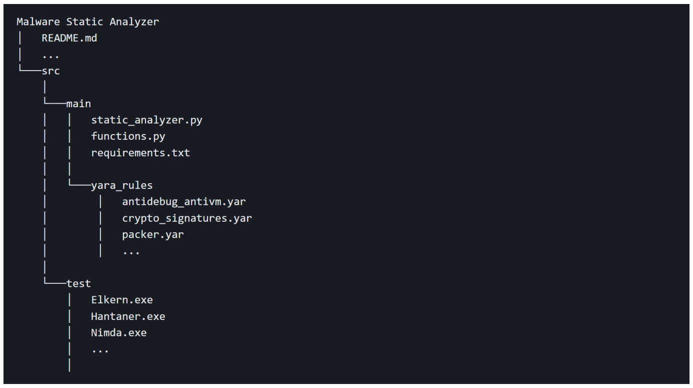
Over the last decade, there has been an 87% increase in malware infections [1]. With such a drastic rate of increase, analysis tools are having a difficult time matching the pace of infection rates. Additionally, many current static analysis tools have a steep learning curve or require a multitude of plugins to be able to use them for general analysis. Within this report, the importance of static analysis as well as a proposed static analysis tool, Static Analyzer, are discussed.

II. Project Description

Static Analyzer analyzes multiple components of a portable executable file including headers, imported DLLs, function calls, etc. Many current static analysis tools require a learning curve or a multitude of plugins to be able to use it for general analysis, however, Static Analyzer is a simple, reliable, and fast analysis tool that aids in the recognition of malicious executable files.

III. Overall Design

A. Project Structure



```
Malware Static Analyzer
│   README.md
│   ...
└───src
    │
    └───main
    │   │   static_analyzer.py
    │   │   functions.py
    │   │   requirements.txt
    │   │
    │   └───yara_rules
    │        │   antidebug_antivm.yar
    │        │   crypto_signatures.yar
    │        │   packer.yar
    │        │   ...
    │
    └───test
        │   Elkern.exe
        │   Hantaner.exe
        │   Nimda.exe
        │   ...
        │
```

Fig. 1. Diagram of overall project structure

B. Program Structure

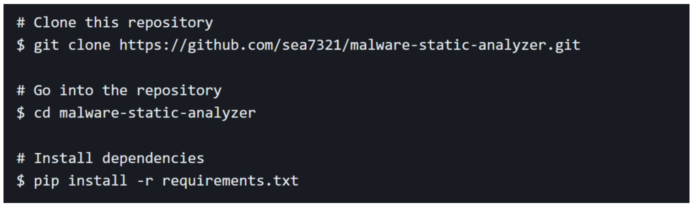
Static_analyzer.py is the main module that analyzes the portable executable files by retrieving file properties, checking for suspicious activity, checking for anti-virtualization features, and using Yara to check for suspicious activity.

Functions.py is a sub module of static_analyzer.py that contains all of the analysis functions for each of the checks in static_analyzer.py.

IV. Installation and Usage

A. Installation

To install Static Analyzer, clone the GitHub repository as shown in Figure 2. Once this is cloned into a desired directory, install all dependencies found in requirements.txt using Pip.



```
# Clone this repository
$ git clone https://github.com/sea7321/malware-static-analyzer.git

# Go into the repository
$ cd malware-static-analyzer

# Install dependencies
$ pip install -r requirements.txt
```

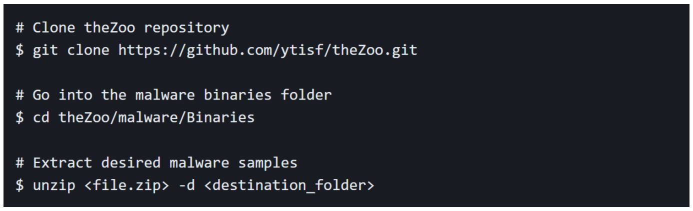
Fig. 2. Installation instructions

B. Obtaining Malware Samples

Once Static Analyzer has been successfully pulled from Github and all dependencies have been installed, malware samples must be acquired to test. To obtain malware samples, clone theZoo's GitHub repository [2] as shown in Figure 3.

Extract all desired malware samples found within /malware/Binaries to the /src/test folder. Note that Static Analyzer requires ".exe" files to be supplied. Folders that were extracted for the analysis portion of this tool are listed below:

- W32.Elkern.B
- W32.HLLP.Hantaner.A
- W32.Nimda.E
- W32.Slammer
- W32.Swen
- Win32.AgentTesla
- Win32.Alina.3.4.B
- Win32.DarkTequila
- Win32.GravityRAT
- Win32.Infostealer.Dexter
- Win32.SofacyCarberp
- Win32.WannaPeace

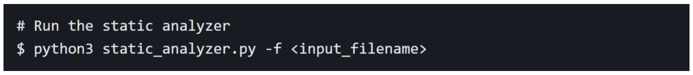


```
# Clone theZoo repository
$ git clone https://github.com/ytisf/theZoo.git

# Go into the malware binaries folder
$ cd theZoo/malware/Binaries

# Extract desired malware samples
$ unzip <file.zip> -d <destination_folder>
```

Fig. 3. Obtaining malware samples instructions

C. Program Execution

To execute Static Analyzer, use the follow command as shown in Figure 4. The input filename should be the path to the malware sample file. If using the instructions above, these malware samples should be in the /src/test folder.



```
# Run the static analyzer
$ python3 static_analyzer.py -f <input_filename>
```

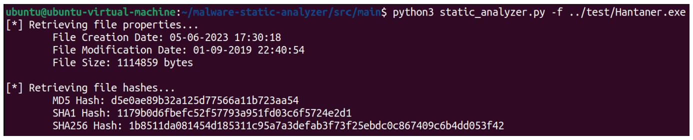
Fig. 4. Program execution instructions

V. Use Case/Example

A. Retrieving File Properties

This section of Static Analyzer retrieves basic file properties from a portable executable file including creation and modification dates, file size, and file hashes.

These can be used to identify files that were recently created/modified or files that have a suspicious file size. Additionally, file hashes can be used to determine the file's authenticity and compare against popular malware file hashes.



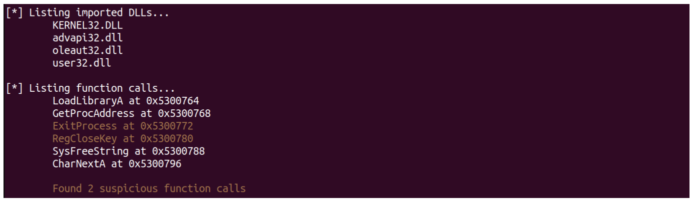
```
ubuntu@ubuntu-virtual-machine:~/malware-static-analyzer/src/main$ python3 static_analyzer.py -f ../test/Hantaner.exe
[*] Retrieving file properties...
        File Creation Date: 05-06-2023 17:30:18
        File Modification Date: 01-09-2019 22:40:54
        File Size: 1114859 bytes

[*] Retrieving file hashes...
        MD5 Hash: d5e0ae89b32a125d77566a11b723aa54
        SHA1 Hash: 1179b0d6fbefc52f57793a951fd03c6f5724e2d1
        SHA256 Hash: 1b8511da081454d185311c95a7a3defab3f73f25ebdc0c867409c6b4dd053f42
```

Fig. 5. File properties sample output for Hantaner.exe

B. Checking Suspicious Activity

The next section of Static Analyzer checks for suspicious activity within the portable executable file such as suspicious API calls or DLL functions. All function calls are listed and compared against common API calls found within malware samples as discussed within the following Windows API Calls article [3].

Within the example output in Figure 6, two of the six function calls that were listed are flagged as suspicious. This serves as a simple and efficient way of determining the activity found within the portable executable file.

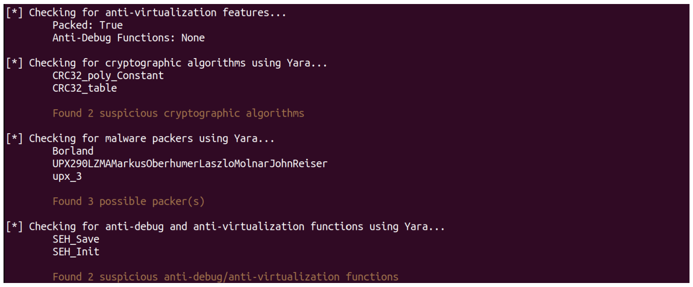


```
[*] Listing imported DLLs...
        KERNEL32.DLL
        advapi32.dll
        oleaut32.dll
        user32.dll

[*] Listing function calls...
        LoadLibraryA at 0x5300764
        GetProcAddress at 0x5300768
        ExitProcess at 0x5300772
        RegCloseKey at 0x5300780
        SysFreeString at 0x5300788
        CharNextA at 0x5300796

        Found 2 suspicious function calls
```

Fig. 6. Suspicious activity sample output for Hantaner.exe

C. Checking for Anti-Virtualization Features

The last section of Static Analyzer checks for anti-virtualization features within the file such as if it is packed or is checking for anti-virtualization functions. Specifically, this section also uses Yara rules downloaded from the Yara-Rules Project GitHub repository [4] to check for cryptographic algorithms, malware packers, and other anti-debug/anti-virtualization functions within the file.



```
[*] Checking for anti-virtualization features...
        Packed: True
        Anti-Debug Functions: None

[*] Checking for cryptographic algorithms using Yara...
        CRC32_poly_Constant
        CRC32_table

        Found 2 suspicious cryptographic algorithms

[*] Checking for malware packers using Yara...
        Borland
        UPX290LZMAMarkusOberhumerLaszloMolnarJohnReiser
        upx_3

        Found 3 possible packer(s)

[*] Checking for anti-debug and anti-virtualization functions using Yara...
        SEH_Save
        SEH_Init

        Found 2 suspicious anti-debug/anti-virtualization functions
```

Fig. 7. Anti-virtualization sample output for Hantaner.exe

VI. Results

As a result of this project, each of the twelve malware samples that were pulled from theZoo's GitHub repository were successfully flagged as suspicious by one or more of the multiple checks performed. This is significant to the static analysis detection community, because Static Analyzer drastically speeds up the analysis processing time in addition to simplifying the output.

VII. Future Work

While this program showed promising results, there are still several ways that it can be improved to increase the functionality and effectiveness such as allowing multiple file formats for analysis, adding additional Yara rules, and enabling multi-threading techniques.

VIII. Conclusion

In conclusion, Static Analyzer is a simple, reliable, and fast analysis tool that aids in the recognition of malicious executable files. Additionally, since this tool contains fundamental techniques to static analysis, it can be easily utilized and modified by anyone within the malware analysis field regardless of skill level.

References

[1] B. Vuleta, "44 must-know malware statistics to take seriously in 2023," Mar 2023. [Online]. Available: https://legaljobs.io/blog/malware-statistics/

[2] "thezoo," https://github.com/ytisf/theZoo, 2014.

[3] "Windows api calls: The malware edition," Apr 2020. [Online]. Available: https://sensei-infosec.netlify.app/forensics/windows/api-calls/2020/04/29/win-api-calls-1.html

[4] "Yara-rules," https://github.com/Yara-Rules/rules, 2014.