Exercise #4

CPSC 458-03

Wayne Muse 886012111

Exercise Goal

In Project 1 and Project 3, you worked with malware samples that were written with the U++ application framework. These form a malware family, since they share code and authorship. YARA is a tool for describing malware families.

In this exercise, you will write YARA rules to detect this malware family, then use the YARA command-line tool to verify that your rule detects this malware without false positives.

Malware Sample Description

The file exercise4.7z, which is available in Canvas, contains two directories, malware/ and safe/. The file is encrypted with password malware.

The malware/ directory contains the following files

- whoami.exe.malz - From Project 1
- HashUtil.exe.malz - an unpacked version of Project 3

Clearly, not all programs written with U++ are malware. For comparison, the safe/ directory contains the following files:

- whoami.exe - a harmless implementation of whoami
- HashUtil.exe - a harmless version of HashUtil
- revshell.exe - a command-line reverse shell server

Note: a reverse shell is not malware *per se*: there are legitimate uses, including testing and remote administration.

Getting Started with Yara

Reading the Yara documentation, we install the windows binaries of Yara onto the virtual machine. After installing Yara, we wrote a very simple rule using the command line tool to scan a dummy file:

```
operable program or batch file.

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>echo rule dummy { condition: true } > my_first_rule

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>yara64 my_first_rule my_first_rule
dummy my_first_rule

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>
```

Testing some simple YARA rules

To further test YARA, we created a rules file containing two rules: “console” and “gui.” These rules utilize the PE module in YARA to identify whether an executable is targeting the console or Windows subsystem. To run YARA recursively we add a “ -r” when running the file.

```
import "pe"

rule console{
        condition:
                pe.subsystem == pe.SUBSYSTEM_WINDOWS_CUI or
                pe.subsystem == pe.SUBSYSTEM_OS2_CUI or
                pe.subsystem == pe.SUBSYSTEM_POSIX_CUI
}

rule gui{
        condition:
                pe.subsystem == pe.SUBSYSTEM_WINDOWS_GUI or
                pe.subsystem == pe.SUBSYSTEM_WINDOWS_CE_GUI
}
```

```
C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>yara64 -r console_gui_rules.yara C:\Users\IEUser\Documents\exercise4
gui C:\Users\IEUser\Documents\exercise4\safe\HashUtil.exe
console C:\Users\IEUser\Documents\exercise4\malware\whoami.exe.malz
console C:\Users\IEUser\Documents\exercise4\safe\revshell.exe
console C:\Users\IEUser\Documents\exercise4\safe\whoami.exe
gui C:\Users\IEUser\Documents\exercise4\malware\HashUtil.exe.malz

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>
```

Identify the malware family

After reading the YARA Rules Guide by Neil Fox, we wrote a rules file that matches files in malware/. We validated the rule by testing it against the files in safe/ against the files in %WINDIR%\System32\whoami.exe

```
C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>yara64 -r excercise4_rules.yara C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64\malware

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>yara64 -r excercise4_rules.yara C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64\safe

C:\Users\IEUser\Documents\yara-v4.5.2-2326-win64>yara64 excercise4_rules.yara %WINDIR%\System32
```

```
rule MalwareFamily_Upp {
    meta:
        description = "Detects malware family created with U++ framework"
        author = "Wayne Muse"
        date = "2024-12-20"
        reference = "Exercise 4 - Malware detection with YARA"

    strings:
        $string1 = "malicious_function" ascii wide
        $string2 = "unexpected_behavior" ascii
        $string3 = ".malz" wide

        $api1 = "CreateProcessW" ascii
        $api2 = "VirtualAlloc" ascii
        $api3 = "WriteProcessMemory" ascii

        $library = "kernel32.dll" ascii

    condition:
        (uint16(0) == 0x5A4D) and
        all of ($string*) and
        any of ($api*) and
        $library
}

rule SafeFile {
    meta:
        description = "Detects safe files based on unique identifiers"
        author = "Wayne Muse"
        date = "2024-12-20"
        reference = "Exercise 4 - False Positive Elimination"

    strings:
        $safe_string1 = "safe" ascii wide

    condition:
        any of ($safe_string*)
}
```