Hands On Lab Unit 5

MICS-252, Fall 2024

Prepared by: Karl-Johan Westhoff

email: kjwesthoff@berkeley.edu

UC Berkleley School of Information

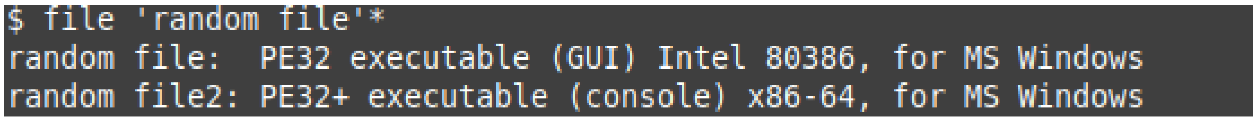
MICS Course 252 Fall 2024 (Kristy Westphal)

1 Reverse engineering of executables

Two files are provided to be subjected to analysis

1.1 ’file’ analysis

Linux has a simple utility for checking a file’ type by its content, see results in Figure 1



```
$ file 'random file'*
random file:  PE32 executable (GUI) Intel 80386, for MS Windows
random file2: PE32+ executable (console) x86-64, for MS Windows
```

Figure 1: *Output from using ’file’ in Linux on ’random_file’ and ’random_file1’*

Results The ’PE32’ and ’PE32+’ indicate the file format is ’Portable Executable’ (the ’+’ indicating version for 64bit memory structure)[1]. The PE32 standard includes some headers in the file so the Windows operating can execute the files, either as a stand alone .exe files or as part of other programs or things running on the OS as .dll etc.[1]

I.e. the files could contain many functions for something running on a Windows OS. The PE file (ELF files for Linux) contain information of how the program should be laid out in memory on the OS.

1.2 Reverse engineering, static analysis using Ghidra

Static reverse engineering includes de-compiling the executables, looking at the code, but not running it. I used ’Ghidra’[**Ghidra**] for the de-compilation.

[1] Windows file extensions for PE’s include: .acm, .ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp, .mun[1]

References

[1] *Portable Executable, Wilipedia*. `https://en.wikipedia.org/wiki/Portable_Executable`. Accessed: 2024-23-13.

[2] *NSA provided opensource reverse engineering tool*. `https://ghidra-sre.org/`. Accessed: 2024-23-13.

Appendices