Reverse Engineering March, 2023

Lab session 0x01

In this lab session we have some black-box binary analysis methods and techniques.

1 Lab files

The files for this lab session are available at https://pwnthybytes.ro/unibuc_re/01-lab-files.zip and the password for the zip file is *infected*.

2 Tools we use (Linux)

In the Linux environment we use two popular tools: strace[1] and ltrace[2]. From the man page and the official web page, these tools serve to:

- strace is a diagnostic, debugging and instructional userspace utility for Linux. It is used to monitor and tamper with interactions between processes and the Linux kernel, which include system calls, signal deliveries, and changes of process state.
- ltrace is a program that simply runs the specified command until it exits. It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process. It can also intercept and print the system calls executed by the program.

The idea is to use these tools to check how a piece of software interacts with external libraries and the kernel (through syscalls[3]). This will give us information about what the software does (files it accesses for read/write, memory, I/O, etc.).

The first step is to install these tools from the command line:

```
$ sudo apt-get install strace ltrace
```

2.1 Tasks

Use strace to understand the syscalls made by a program. For example, run:

```
$ sudo strace ls .
```

Check some of the syscalls: execve, open, close, fstat, ioctl, read, write, exit, mmap, munmap, etc.

Use ltrace to understand what is a good input for the *obscure* binary from the zip file.

3 Tools we use (Windows)

For the Windows operating system, a tool like Task Manager, while useful in general, does not provide enough detailed information about a process. We therefore introduce two new tools: Process Monitor[4] and API Monitor[5]. From their official web pages, these tools serve to:

[1] https://strace.io/
[2] https://man7.org/linux/man-pages/man1/ltrace.1.html
[3] https://man7.org/linux/man-pages/man2/syscalls.2.html
[4] https://docs.microsoft.com/en-us/sysinternals/downloads/procmon
[5] http://www.rohitab.com/apimonitor

- Process Monitor is an advanced monitoring tool for Windows that shows real-time file system, Registry and process/thread activity. It combines the features of two legacy Sysinternals utilities, Filemon and Regmon, and adds an extensive list of enhancements including rich and non-destructive filtering, comprehensive event properties such as session IDs and user names, reliable process information, full thread stacks with integrated symbol support for each operation, simultaneous logging to a file, and much more. Its uniquely powerful features will make Process Monitor a core utility in your system troubleshooting and malware hunting toolkit.
- API Monitor is a free software that lets you monitor and control API calls made by applications and services. Its a powerful tool for seeing how applications and services work or for tracking down problems that you have in your own applications.

4 Lab tasks

4.1 Task 1: for Linux

Check the *crackme* binary from the zip file. The binary will ask for your input and apply some checks on it. Whenever a check is failed, it will print "WRONG". Since the binary is heavily obfuscated, static analysis is out of the question.

User ltrace/strace and any other tools to solve the following tasks:

- use pwntools[6] and Python to automatically call the binary directly and get its output; (1p)
- stop calling the binary directly; wrap it inside ltrace and get all the library functions called; (1p)
- bypass the length check by trying various inputs; (2p)
- pass all the other checks; (2p)
- find the correct password. (2p)

4.2 Task 2: for Windows

Check the *malware.exe* binary from the zip file. Using API Monitor and ProcMon, gather as much information as possible about the malware. Answer the following questions:

- where does it connect to? (2p)
- what registry keys does it access and why? (2p)

5 Bonus task

In real-world infection scenarios, malware authors try to make their "software" as robust as possible. To this end, various synchronization mechanisms must be put into place. However, the reverse can be applied by defenders: the same synchronization mechanisms can be put into place or simulated in order to fool the malware into thinking that the machine is already infected. To get the bonus points, figure out how this malware uses synchronization to avoid reinfection and then devise a way to "vaccinate"[7] machines against this malware. (4p)

[6] https://docs.pwntools.com/en/stable/

[7] https://www.sans.org/blog/looking-at-mutex-objects-for-malware-discovery-and-indicators-of-compromise/