Work & documentation notes of various the Leviathan wargame

Galen Rowell

July 30, 2020

Leviathan

leviathan0

Password to enter: *leviathan0*
Challenge: Within a hidden folder inside the home directory, there was a *bookmarks.html* file. With a quick visual inspection the password is listed within the file. The file is long, and a more suitable method for anything larger or more complex would be a regex search with **grep**.

```
grep 'password' bookmarks.html
```

leviathan1

Password to enter: *rioGegei8m*
Challenge: This level provides a Linux executable which, with the correct password, launches us into a shell of the next leviathan level. From there we can read the password of *leviathan2*.

The shell command **file** is used to test the encoding & file-type of a given file, which is particularly useful on binaries & executable files. The latter part of **file**'s output *”not stripped”* informs us that the debugging symbols were included in this last compilation. This is particularly useful as it allows us to easily trace the given file.

Various debugging and executable-tracing commands exist, such as **gdb, strace, ltrace & sysdig**. **ltrace** is fantastic tool which aims at tracing the execution of a given executable, with particular focus on library calls. **strace** is comparison similar to **ltrace**, except with a heavier focus upon system calls.

With these two commands, one can see **line #12** shows the password for the executable.

the shell during reversal

```
1  file ./check
2    check: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
3    dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32,
4    BuildID[sha1]=c735f6f3a3a94adcad8407cc0fda40496fd765dd, not stripped
5  ltrace ./check
6    __libc_start_main(0x804853b, 1, 0xffffd774, 0x8048610 <unfinished ...>
7    printf(”password: ”)                              = 10
8    getchar(1, 0, 0x65766f6c, 0x646f6700password: testPassword
9    )                    = 116
10   getchar(1, 0, 0x65766f6c, 0x646f6700)             = 101
11   getchar(1, 0, 0x65766f6c, 0x646f6700)             = 115
12   strcmp(”tes”, ”sex”)                              = 1
13   puts(”Wrong password, Good Bye ...”Wrong password, Good Bye ...
14   )                     = 29
15   +++ exited (status 0) +++
```

Note: *Line #8: ”testPassword” was manually entered*
Note: *Line #12: the executable checks our input with the string ”sex”, the password for the script*

leviathan2

Password to enter: *ougahZi8Ta*
Challenge: **ltrace** is a fantastic tool for discovering the exploit here, it's use reveals two important function calls.

```
access(”filename”, 4)
```

```
system(”/bin/cat filename”file content
```

Note: *'file content' is output by* ***ltrace*** *as part of the executable examination*

These two functions check for read permissions of the file and pass the filename to the command line receptively. The flaw in the script lies in the quotations, the double quotes around the filename are dropped for the shell call.

This causes issues with a spaced filename, as **cat** will print out each argument given. This can be taken advantage of with the following:

```
echo ”file a” > a
echo ”file a b” > 'a b'
ln -s /etc/leviathan_pass/leviathan3 b
~/printfile ”a b”
```

Note: *Files may need to be given read access to the others group*

leviathan3

Password to enter: *Ahdiemoo1j*
Challenge:

Links & resources

1. When scripting, it is often useful to have a temporary directory where files can be created & modified without the risk of littering such files about the filesystem. So a temporary directory (often in /tmp/) is useful, **mktemp** does this:

   move to the new temporary directory

   ```
   cd $(mktemp -d)
   ```

   store the new temporary directory path

   ```
   tmp_dir=$(mktemp -d)
   ```