1 Format-string 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 command explained in the `readme`:

```
$ sudo sysctl -w kernel.randomize_va_space=0
```

1.2 Task 1

You can use the provided `build_string.py` Python script to write a new one that puts values on the stack. In particular, to exploit the format-string vulnerability of the target program, we can write an arbitrary number of %s on the stack until we make it crash. You can execute our solution as follows:

```
$ ./exploit1.py 10 && cat badfile | nc 10.9.0.5 9090
```

1.3 Task 2a

As in the previous task, we can use the `build_string.py` Python script to create an input message having four hard-coded characters at the start, followed by as many %x characters as required to print the first four ones. The number of %x is at least 54 since the difference between the addresses of the frame pointer and of the input buffer, printed on the terminal, is 54 bytes. The exploit can be executed as follows:

```
$ ./exploit2a.py 64 && cat badfile | nc 10.9.0.5 9090
```

1.4 Task 2b

To print the content of the secret message located at the address 0x080b4008, we define a message having at first the address, then N %x and a final %s, used to print the content located at 0x080b4008. The exploit can be executed as follows:

```
$ ./exploit2b.py 64 && cat badfile | nc 10.9.0.5 9090
```

1.5 Task 3a

The task is very similar to the previous one, but we start the crafted message with the target address first. Then, we add N %x and a final %n, which allows to write a value on the stack. The new value saved into the target variable corresponds to the number of characters printed out by the `printf` function. The exploit can be executed as follows:

```
$ ./exploit3a.py 64 && cat badfile | nc 10.9.0.5 9090
```

1.6 Task 3b

Similarly to the previous task, we have now to write on the stack a specific value, which is equal to 0x5000. We can exploit the same idea used in Task 3a, keeping in mind that the target variable will contain the

number of characters printed out by the `printf` function. Thus, we can use the last %x to get a counter to 0x5000 and save the counter value on the stack. The exploit can be executed as follows:

```
$ ./exploit3b.py 64 && cat badfile | nc 10.9.0.5 9090
```
