Proactive Computer Security
Assignment 5

Nikolaj Dybdahl Rathcke (rfq695)

June 3, 2016

1 Fuzzer

The fuzzer is found `fuzzer.py`. The fuzzer I have implemented is a very simple one and the approach for its construction is very simple as well. I started by making a random byte string of a fixed length and then slowly increasing (by 1) the length of it if it did not cause any crash. The string is always stripped for new line bytes, `NULL` bytes and spaces as I suspected these might mess up the fuzzing. It would send the random byte string through all the available commands and when it started causing crashes, I used the process of elimination to realize my crash came from the `JOINING` command. Obviously, the crash could be caused by other commands as well, but I knew this one worked. Thus, all the fuzzer does is creating a random byte string of length 1000 and then incrementing the length by one until it crashes (random string for each iteration), that is, until a `core` file is produced. I have commented the fuzzer, so it should make sense as it's also very simple.
Note that the connected is closed and started for each iteration in order to join a channel again. If the connection is closed for some reason, we reestablish it as well.

2 Findings

Usually, the fuzzer has to run for under 50 iterations before crashing. The line that succeeded to crash it and make a `core` file is written to a file `ded.txt` (you can toggle the `autowin` variable if you want to see the fuzzer in action as right now it just uses one that worked located in `ded2.txt`).
If we fire up `gdb` on the core produced by the string in `ded2.txt` (it's called `core2` because the fuzzer runs until there is a file named `core` in the path):

```
$ gdb yairc core2
```

We can see that it has a segmentation fault at `0x15ad13fe` (if we use the line in `ded2.txt`). If we take a look at the input we gave and use a little time on this, we can see that it actually the address actually correspond to the bytes on indices 1005 to 1008 (which we can verify by running it again). Thus, we can change this to wherever we want in the memory by constructing a string that crashes the thread with the desired address on these indices of the string.