Exploit dot education Phoenix notes

`ssh -p2222 user@localhost` with password "user"

There is also the root:root account

`scp -p2222 user@localhost:/opt/phoenix/amd64/stack-zero ~/Documents/pheonix/`

Stack Zero

Aim: To change the contents of the changeme variable

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define BANNER \
  "Welcome to " LEVELNAME ", brought to you by https://exploit.education"

char *gets(char *);

int main(int argc, char **argv) {
  struct {
    char buffer[64];
    volatile int changeme;
  } locals;

  printf("%s\n", BANNER);

  locals.changeme = 0;
  gets(locals.buffer);

  if (locals.changeme != 0) {
    puts("Well done, the 'changeme' variable has been changed!");
  } else {
    puts(
        "Uh oh, 'changeme' has not yet been changed. Would you like to try "
        "again?");
  }

  exit(0);
}
```

Notes:

A **Struct** in C is a collection of variables under one name (basically an object)

`char *gets(char *);` is the function prototype, it just defines the syntax for the gets function, I have no idea why it's in this main file and not in the imported library. removing it did not affect the challenge, but did give me an "implicit declaration" warning. **Edit:** on further research, I get this warning because gets was removed from standard libraries due to being unsafe

The file type we are working on is `stack-zero: setuid, setgid ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /opt/phoenix/x86_64-linux-musl/lib/ld-musl-x86_64.so.1, not stripped`

Answer

Explanation

`gets(locals.buffer);` reads from stdin, and then stores the value in the variable called "buffer" inside the struct named "locals".

by entering an input of > 64 bytes, you overflow the allocated buffer

because the next space in memory is the changeme variable (it was declared just after buffer), the overflow data spills into it

This is why when we use gdb, we can see the variable changeme has "q" (0×71) with the input "aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaq", q is the 65th byte

Process:

- gdb -d stack-zero
- disassemble main
- break *<address just before the jump>
- info registers eax
- the value will be 0×71 :)