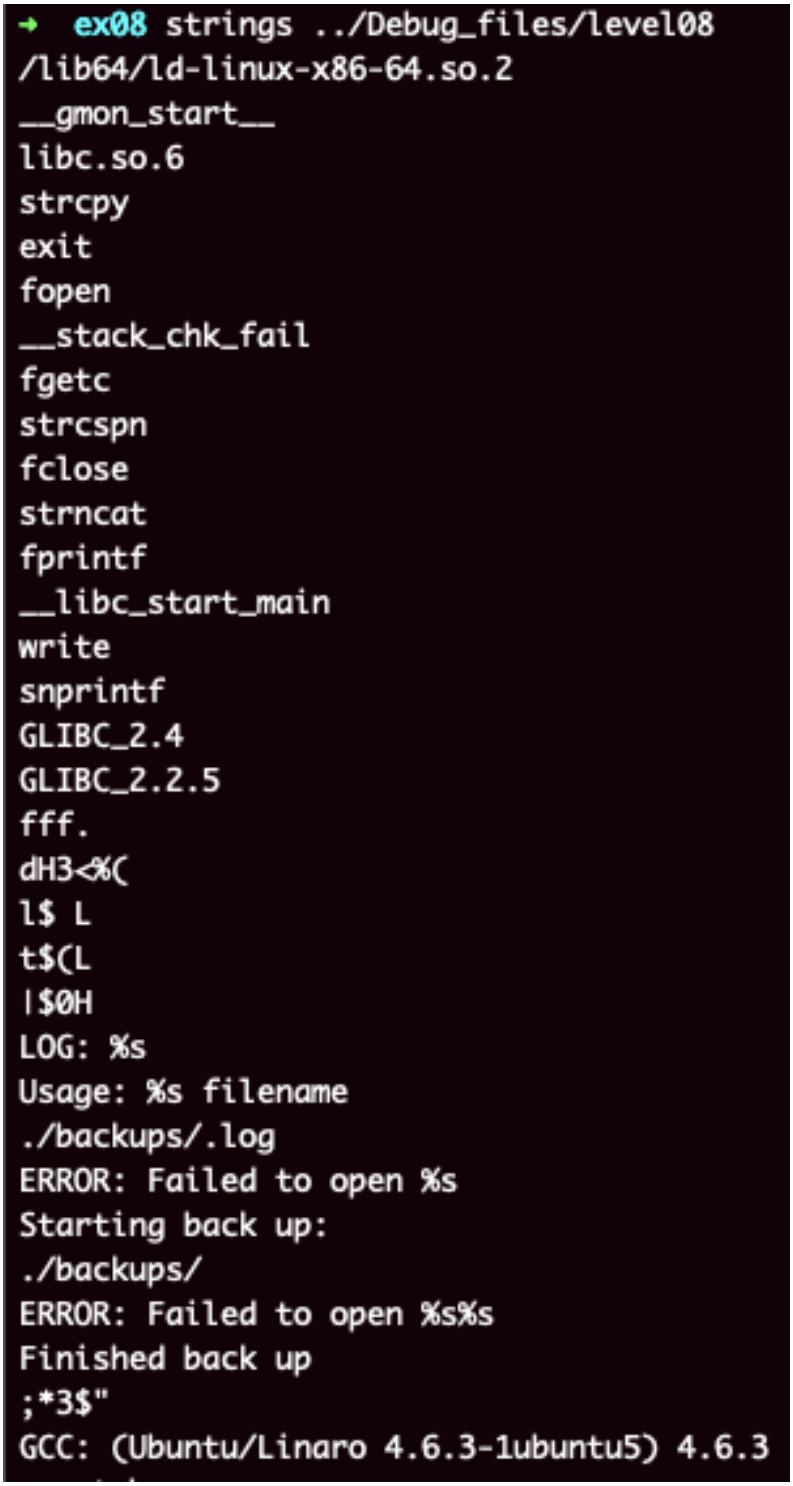
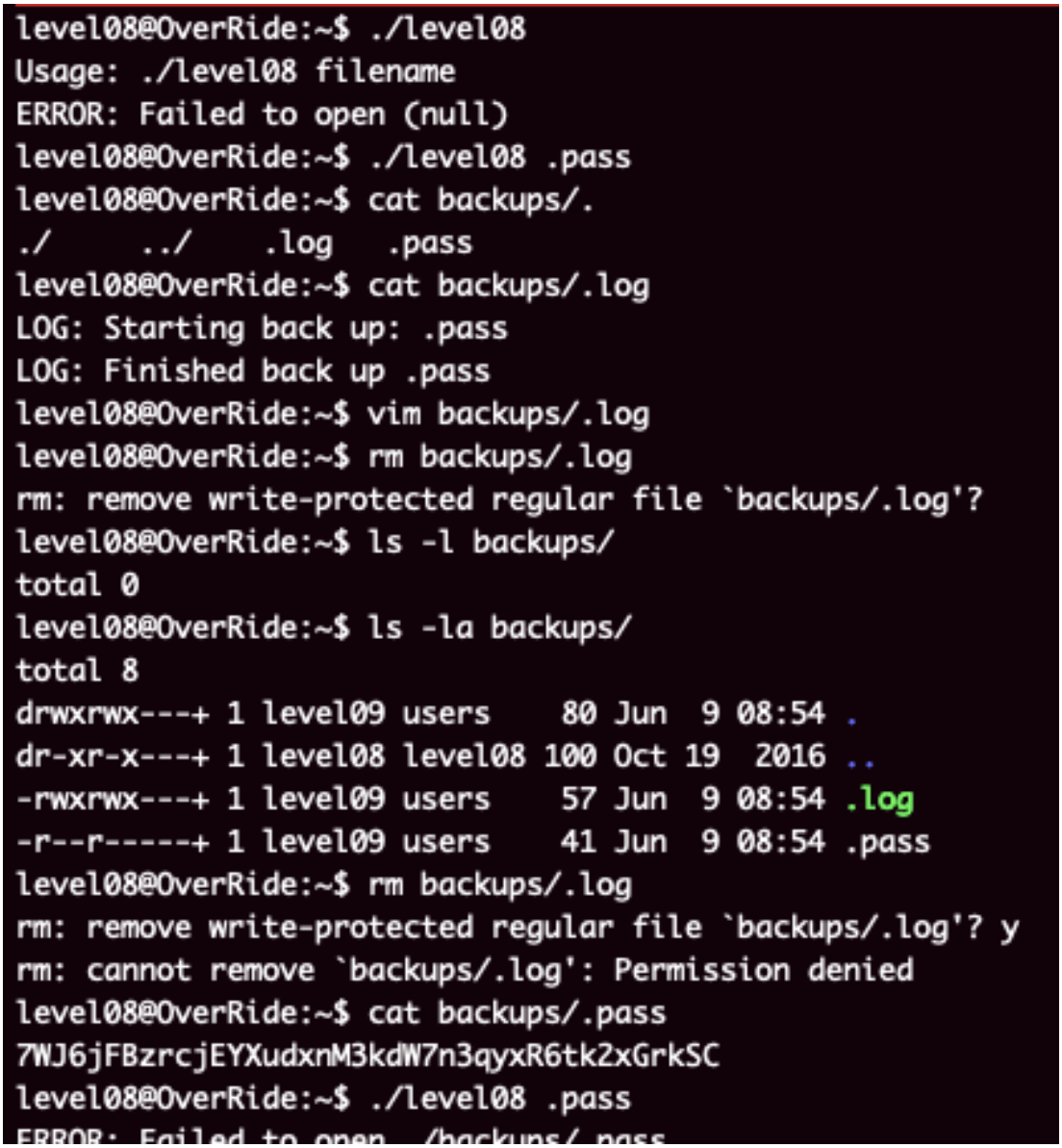
LEVEL08:

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x4007e0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          8496 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 26
```


➜ ex08 strings ../Debug_files/level08
/lib64/ld-linux-x86-64.so.2
__gmon_start__
libc.so.6
strcpy
exit
fopen
__stack_chk_fail
fgetc
strcspn
fclose
strncat
fprintf
__libc_start_main
write
snprintf
GLIBC_2.4
GLIBC_2.2.5
fff.
dH3<%(
l$ L
t$(L
|$0H
LOG: %s
Usage: %s filename
./backups/.log
ERROR: Failed to open %s
Starting back up:
./backups/
ERROR: Failed to open %s%s
Finished back up
;*3$"
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3



The program expect one argument, to be a file he can access or open, copy it in backups directory

For this exercise, I took a look at ghidra to make a first idea:

```
void log_wrapper(FILE *stream_back,char *sentence,char *first_str_main)

{
  size_t size_before_\n;
  ulong compteur;
  ulong compteur2;
  long in_FS_OFFSET;
  byte direction_flag;
  undefined8 backup_log_stream;
  char save_sentence [264];
  long stack_protect;
  char c;
  char *save;

  direction_flag = 0;
  stack_protect = *(long *)(in_FS_OFFSET + 0x28);
  backup_log_stream = stream_back;
  strcpy(save_sentence,sentence);
  compteur = 0xffffffffffffffff;
  save = save_sentence;
  do {
    if (compteur == 0) break;
    compteur = compteur - 1;
    c = *save;
    save = save + (ulong)direction_flag * -2 + 1;
  } while (c != '\0');
  compteur2 = 0xffffffffffffffff;
  save = save_sentence;

  do {
    if (compteur2 == 0) break;
    compteur2 = compteur2 - 1;
    c = *save;
    save = save + (ulong)direction_flag * -2 + 1;
  } while (c != '\0');
  snprintf(save_sentence + (~compteur2 - 1),0xfe - (~compteur - 1),first_str_main);
  size_before_\n = strcspn(save_sentence,"\n");
  save_sentence[size_before_\n] = '\0';
  fprintf(backup_log_stream,"LOG: %s\n",save_sentence);
  if (stack_protect != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}
```

```
uint64_t main(int ac,char **av)

{
  char cVar1;
  int __fd;
  int i;
  FILE *backup_log_stream;
  FILE *param_stream;
  ulong compteur;
  uint64_t *addr_name_backup;
  long in_FS_OFFSET;
  byte direction_flag;
  char c;
  uint64_t name_backup_dir;
  undefined2 name_backup_dir_bis;
  char name_backup_dir_ter;
  long stack_protect;

  direction_flag = 0;
  stack_protect = *(long *)(in_FS_OFFSET + 0x28);
  c = -1;
  if (ac != 2) {
    printf("Usage: %s filename\n",*av);
  }
  backup_log_stream = fopen("./backups/.log","w");
  if (backup_log_stream == (FILE *)0x0) {
    printf("ERROR: Failed to open %s\n","./backups/.log");
                    /* WARNING: Subroutine does not return */
    exit(1);
  }
```

```
log_wrapper(backup_log_stream,"Starting back up: ",av[1]);
param_stream = fopen(av[1],"r");
if (param_stream == (FILE *)0x0) {
  printf("ERROR: Failed to open %s\n",av[1]);
                    /* WARNING: Subroutine does not return */
  exit(1);
}
name_backup_dir = 0x70756b6361622f2e;
name_backup_dir_bis = 0x2f73;
name_backup_dir_ter = '\0';
compteur = 0xffffffffffffffff;
addr_name_backup = &name_backup_dir;
do {
  if (compteur == 0) break;
  compteur = compteur - 1;
  addr_name_backup = (uint64_t *)((long)addr_name_backup + (ulong)direction_flag * -2 + 1);
  cVar1 = *(char *)addr_name_backup;
  addr_name_backup = addr_name_backup;
} while (cVar1 != '\0');
                    /*  O_CREATE | O_EXCL | O_WRONLY */
strncat((char *)&name_backup_dir,av[1],99 - (~compteur - 1));
__fd = open((char *)&name_backup_dir,0b11000001,0x1b0);
if (__fd < 0) {
  printf("ERROR: Failed to open %s%s\n","./backups/",av[1]);
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

```
while( true ) {
  i = fgetc(param_stream);
  c = (char)i;
  if (c == -1) break;
  write(__fd,&c,1);
}
log_wrapper(backup_log_stream,"Finished back up ",av[1]);
fclose(param_stream);
close(__fd);
if (stack_protect != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
return 0;
```

```
GNU_STACK       0x0000000000000000 0x0000000000000000 0x0000000000000000
                0x0000000000000000 0x0000000000000000  RWE    8
```

I observe a vulnerability in :

- snprintf() in log_wrapper: because our first program argument is the ***format str*** of the function:
    1- We could want to overwrite the address of :
        - fopen() (which is the next extern function)
        - EIP (main)

2- By the address of:

- system(), which means we would need to write twice 32bits:

address of system() and address of '/bin/sh', 4 octet further.

- a shellcode stored in a ENV VAR (we wouldn't take the risk to give it as a program argument)

POTENTIAL PROBLEM: max size

Target address : 0x7fffffffe510

Destination address : 0x7fffffff e8db

0x7fffff ffe8 db

- 0x7fffffffe510: 0xdb = 219 - 24 = 195
- 0x7fffffffe511: 0xffe8 - 0xdb = 65293
- 0x7fffffffe513: 0x7fffff - 0xffe8 = 8323095

This is compiled with *-fno-stack-protector,* otherwise s[] overflow and it is detected by the canary...

```
char s[264];
int i = 0;
int j = 0;
int k = 0;

snprintf(s, 240, "%219c%2$n%65293c%3$n%8323095c%4$n", 'a', &i, &j, &k);
printf("%s\n%016X\n%016X\n%016X\n", s, k,  j, i);
return 0;
```

```
00000000007FFFFF
000000000000FFE8
00000000000000DB
```

Let's try in gdb

- We can try to override the argument of the open of our copy : './backups/.X':

1-

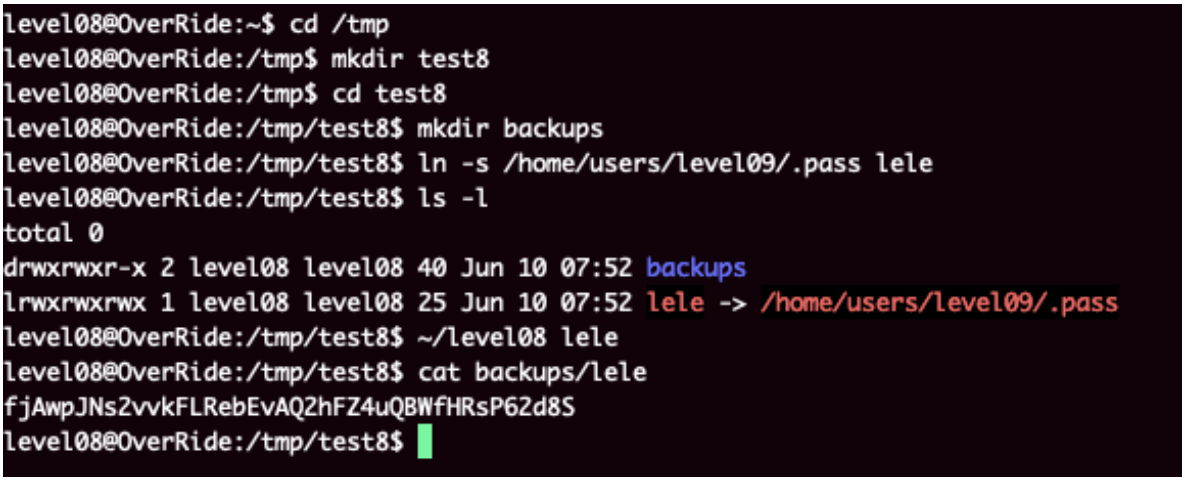
None of them were possible.
But I haven't realize but actually when I give a file argument that include '/', it compose a absolute path that fopen rejects:

So the target file is '/home/users/level09/.pass', actually the only error is because of the '/', in the open of my './backups//home/users/level09/.pass'

Even with my .pass file given in absolute path, it doesnt work for the reason above.

I am not sure that fopen will work but it should, let's try with a symbolic link:

It works !!



Flag: fjAwpJNs2vvkFLRebEvAQ2hFZ4uQBWfHRsP62d8S