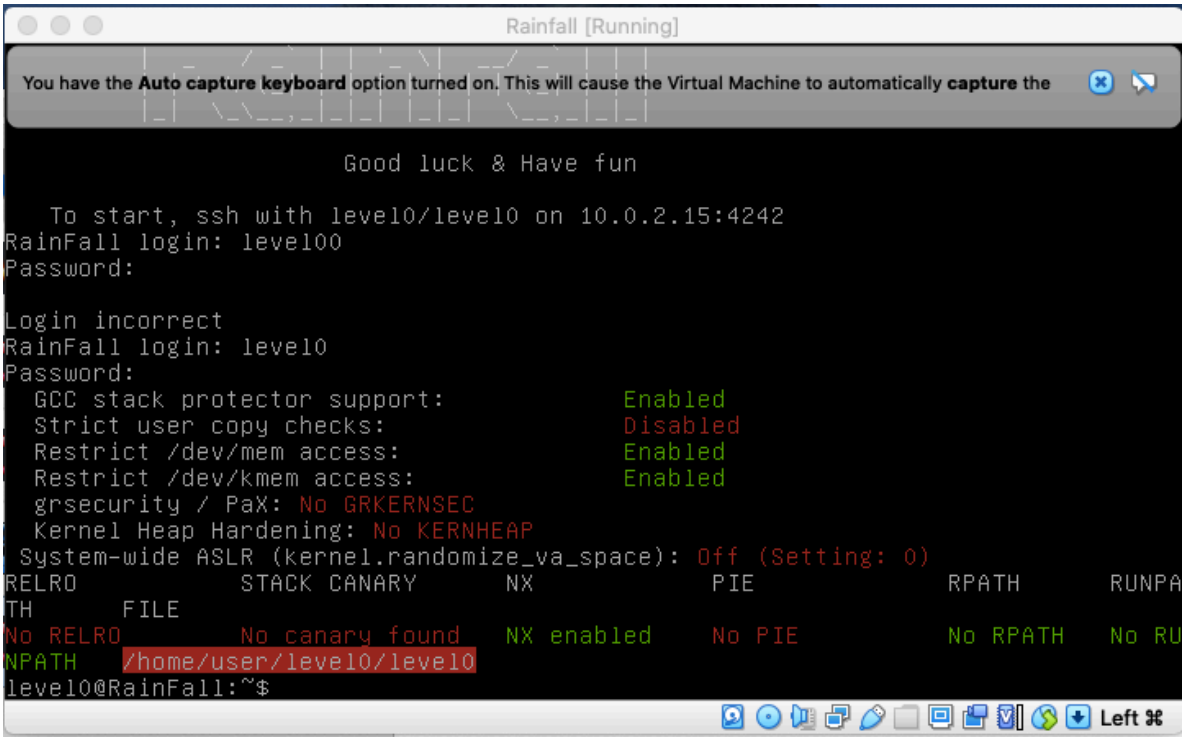
LEVEL0:

login: level0

passwd: level0

After logging in, I have these informations on the screen: (about the iso)



First, I check online for grsecurity and other keywords:

- Grsecurity : Grsecurity® is an extensive security enhancement to the Linux kernel that defends against a wide range of security threats
  https://grsecurity.net/

- GCC stack protector support: Stack protection will abort your program if it detects overwrites of the return address or similar portions of the stack
  https://stackoverflow.com/questions/1629685/when-and-how-to-use-gccs-stack-protection-feature

- KERNHEAP:
  3 - Integrity assurance for kernel heap allocators

3.1 - Meta-data protection against full and partial overwrites
3.2 - Detection of arbitrary free pointers and freelist corruption
3.3 - Overview of NetBSD and OpenBSD kernel heap safety checks
3.4 - Microsoft Windows 7 kernel pool allocator safe unlinking

- ASLR: Address space layout randomization (ASLR) is a memory-protection process for operating systems (OSes) that guards against buffer-overflow attacks by randomizing the location where system executables are loaded into memory.
  https://www.techtarget.com/searchsecurity/definition/address-space-layout-randomization-ASLR

- PIE: position indépendant executable In computing, **position-independent code**[1] (**PIC**[1]) or **position-independent executable** (**PIE**)[2] is a body of machine code that, being placed somewhere in the primary memory, executes properly regardless of its absolute address.

Now our iso is turning, I'll connect via ssh.
And now I have done analyses of what was in front of my eyes, let's dig around to see where we are and how to evolve through sessions, escalating rights more or less. I look around and capture interesting things.
A simple 'ls' will show me what ressources I have. It outputs me a binary name level0, owned by level1 (which is supposed to be the uid of the process), and to group users. Let's check if I am in that group.

```
level0@RainFall:~$ ls -l
total 732
-rwsr-x---+ 1 level1 users 747441 Mar  6  2016 level0
level0@RainFall:~$ id
uid=2020(level0) gid=2020(level0) groups=2020(level0),100(users)
level0@RainFall:~$ groups
level0 users
level0@RainFall:~$
```

The subject says:

- Une fois connecté, vous allez devoir trouver le moyen permettant de lire le fichier `.pass` avec le compte utilisateur "levelX" du niveau suivant (X = numéro du niveau suivant).
- Ce fichier `.pass` est situé à la racine du home de chaque utilisateur (level0 exclu).

I assume that executing bin level0, or exploiting it will give me access to the session of level1, in which a '.pass' file will be present. Once inside level1 account, I guess the subjects mean that I will need the permissions of the next level to be able to read the '.pass' file, itself will give me the password, I guess, of the next session level.
But now, my first step is to find a way of changing session to level1. And I'll dig all the clue I'll encounter. I suppose bin level0 is the .pass file of its session.

I will follow this process to look around :
1- Execution level0:

```
level0@RainFall:~$ ./level0
Segmentation fault (core dumped)
level0@RainFall:~$ ./level0 1
No !
level0@RainFall:~$ ./level0 1 2
No !
level0@RainFall:~$ ./level0 1 2 3
No !
level0@RainFall:~$ ./level0
Segmentation fault (core dumped)
level0@RainFall:~$
```

2- Analyze binary level0:

strings level0: Can't find yet online explanations

```
level0@RainFall:~$ strings level0
-bash: /usr/bin/strings: Input/output error
level0@RainFall:~$
```

objdump -d level0 / gdb level0:

I just read those lines, and try to launch : ./level0 423 because I just saw that the first call was an atoi of the first argument, and a check if the result was 0x1a7 (423), and it gave me that:

```
Dump of assembler code for function main:
   0x08048ec0 <+0>:     push   %ebp
   0x08048ec1 <+1>:     mov    %esp,%ebp
=> 0x08048ec3 <+3>:     and    $0xfffffff0,%esp
   0x08048ec6 <+6>:     sub    $0x20,%esp
   0x08048ec9 <+9>:     mov    0xc(%ebp),%eax
   0x08048ecc <+12>:    add    $0x4,%eax
   0x08048ecf <+15>:    mov    (%eax),%eax
   0x08048ed1 <+17>:    mov    %eax,(%esp)
   0x08048ed4 <+20>:    call   0x8049710 <atoi>
   0x08048ed9 <+25>:    cmp    $0x1a7,%eax
```

```
level0@RainFall:~$ ./level0 423
$ whoami
level1
$
```

Let's see what to do next now.

I'll dig a bit more on the direction of bin level0

https://stackoverflow.com/questions/61907360/how-the-

compiler-does-data-binding-during-compile-time

Reverse:

```
int main(int ac, char **av)
{
    char *s = argv[1];
    int arg = atoi(s);

    if (arg != 423)
    {
        write(1, « No ! », len());
        return 0;
    }
    char *s1 = strdup(« /bin/sh »);
    gid_t egid = getegid();
    uid_t euid = geteuid();
    setresgid(egid, egid, egid);
    setresuid(euid, euid, euid);
    execve(« /bin/sh », s);
}
```

Ok we are now in a subshell process which the uid is level1.
There is a '.pass' file at the ~ that contains
1fe8a524fa4bec01ca4ea2a869af2a02260d4a7d5fe7e7c24d8617e6dca12d3a.
I guess it is the password of the level1 account.
Let's try:

```
level0@RainFall:~$ ./level0 423
$ ls -la
ls: cannot open directory .: Permission denied
$ pwd
/home/user/level0
$ cd ~
/bin/sh: 3: cd: can't cd to /home/user/level0
$ cd /home/user/level1
$ ls -la
total 17
dr-xr-x---+ 1 level1 level1   80 Mar  6  2016 .
dr-x--x--x  1 root   root    340 Sep 23  2015 ..
-rw-r--r--  1 level1 level1  220 Apr  3  2012 .bash_logout
-rw-r--r--  1 level1 level1 3530 Sep 23  2015 .bashrc
-rwsr-s---+ 1 level2 users  5138 Mar  6  2016 level1
-rw-r--r--+ 1 level1 level1   65 Sep 23  2015 .pass
-rw-r--r--  1 level1 level1  675 Apr  3  2012 .profile
$ cat .pass
1fe8a524fa4bec01ca4ea2a869af2a02260d4a7d5fe7e7c24d8617e6dca12d3a
$ ^C
$
level0@RainFall:~$ su level1
Password:
RELRO           STACK CANARY      NX            PIE             RPA
TH      RUNPATH      FILE
No RELRO        No canary found   NX disabled   No PIE          No
RPATH   No RUNPATH   /home/user/level1/level1
level1@RainFall:~$
```

Flag:
1fe8a524fa4bec01ca4ea2a869af2a02260d4a7d5fe7e7c24d8617e6dca12d3a