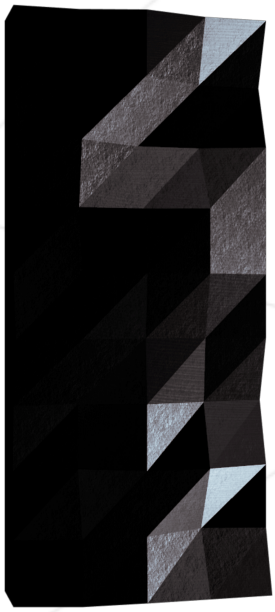

Project SECURITY

Over Ride

42 Staff pedago@staff.42.fr

Summary: This project follows the RainFall project. It will teach you how to exploit the (elf-like) binary.

Contents

Chapter I

Preamble

There is something wrong..

Chapter II

Introduction

As a developer, you might have to work on softwares that will be used by hundreds of persons.

You have learned to develop more or less complex programs without taking security into account.

With this project, you will quickly realize it's rather easy to exploit issues that can be very easily avoided.

Once you're through with this project, you will have a clearer understanding of the RAM. And this will really help you design a bugless program!

Chapter III

Objectives

This project aims to further your knowledge in the world of elf-like binary exploitation in in i386 system.

The more or less complex methods you will use will gibe you a new perspective on IT in general but mostly raise your awareness on issues coming from programming common malpractice.

You will be challenged during this project. You have to overcome these challenges by yourself. The way you'll be dealing with these challenges must be yours and YOURS ONLY. The point is to help you develop some logic and acquire reflexes that will help you all along your career. Before asking for help, ask yourself if you have factored all the possibilities in.

Chapter IV

General instructions

- This project will only be evaluated by humans.
- You may have to prove your results during the evaluation. Be ready!
- To make this project, you will have to use a VM(64 bits). Once you have started your machine with the ISO provided with this subject, if your configuration is right, you will get a simple prompt with an IP:

If the IP address is not visible, you will get it with the command ifconfig once you're connected.

- Then, you will be able to connect using the following couple of `login:password`: `level00:level00`.

  You really should use the SSH connection available on port 4242:

  ```
  $> ssh level00@192.168.1.13 -p 4242
  ```

- Once logged-in, you will have to find a way to read the ".pass" file with the "next level level0X" (X = numéro next level) account.
- This ".pass" file is located at the home root of each (level00 exclu) user.

- Of course, once you've reached level9 you will have to go towards the end user.

- Here is a session example:

```
level00@OverRide:~$ ./level00 $(exploit)
$ cat /home/users/level01/.pass
???????????????????????
$ exit
level00@OverRide:~$ su level01
Password:
level01@OverRide:~$ _
```

- Nothing is left to chance. If there is a problem, start wondering if your code is not the cause.

- Using an automation tool is cheating. Cheating gets you a -42.

- Of course, in case of a true bug, run to the educational team!

- You can post your questions on the forum, Jabber, IRC, Slack...

Chapter V

Mandatory part

- Your repo must include anything that helped you solve each validated test.
- Your repository will look like this:

```
$> ls -al
[..]
drwxr-xr-x 2 root root 4096 Dec 3 XX:XX level00
drwxr-xr-x 2 root root 4096 Dec 3 XX:XX level01
drwxr-xr-x 2 root root 4096 Dec 3 XX:XX level02
drwxr-xr-x 2 root root 4096 Dec 3 XX:XX level03
[..]
$> ls -alR level00
level0:
total 16
drwxr-xr-x 3 root root 4096 Dec 3 15:22 .
drwxr-xr-x 6 root root 4096 Dec 3 15:20 ..
-rw-r--r-- 1 root root  5 Dec 3 15:22 flag
-rw-r--r-- 1 root root 50 Dec 3 15:22 source
drwxr-xr-x 2 root root 4096 Dec 3 15:22 Ressources

level0/Ressources:
total 8
drwxr-xr-x 2 root root 4096 Dec 3 15:22 .
drwxr-xr-x 3 root root 4096 Dec 3 15:22 ..
-rw-r--r-- 1 root root  0 Dec 3 15:22 whatever.wahtever
$> cat level00/flag | cat -e
XXXXXXXXXXXXXXXXXXXXXXXXXXXX$
$> nl level00/source
1   #include <stdio.h>
2   int
3   main(void) {
4   printf("Code, source!\n");
5       return (0x0);
6   }
$> _
```

- You will keep everything you need to prove your results during the evaluation in the Resource folder. The `flag` file may be empty, but you may have to explain why.
- The source file must only include the exploited binary in a form any developer could understand. You're free to choose your language.

WARNING: You must be able to clearly and precisely explain anything that is included in the folder. The folder mustn't include ANY binary.

- If you need to use a specific file that's included on the project's ISO, you must download it during the evaluation. You must put it in your repo under no circumstances.
- If you plan to use a specific external software, you must set up a specific environment (VM, docker, Vagrant).
- You're invited to create scripts that will make you stall, but you will have to explain them during the evaluation.
- For the mandatory part, you must complete the following list of levels:
  - level00.
  - level01.
  - level02.
  - level03.
  - level04.
  - level05.
  - level06.
  - level07.
  - level08.
- During the evaluation, each member of the group must be able to justify each challenge solved:

Hey, smarty (or not so smarty) pants! You cannot bruteforce the ssh flags. This would be useless anyway, since you will have to justify your solution during the evaluation.

Chapter VI

Bonus part

Bonus will be taken into account only if the mandatory part is PERFECT. PERFECT meaning it is completed, that its behavior cannot be faulted, even because of the slightest mistake, improper use, etc... Practically, it means that if the mandatory part is not validated, none of the bonus will be taken in consideration.

For the bonus part, you can complete the following list of levels:

- level09

The last user is "end". Becoming "root" is not a bonus.

Chapter VII

Turn-in and peer-evaluation

As usual, turn in your work on your repo `GiT`. Only the work included on your repo will be reviewed during the evaluation.

`Exceptionally, this project will be evaluated by a member of the educational team. When the project is closed and you want to evaluate it, contact ??? (??? on Slack) to arrange a meeting.`