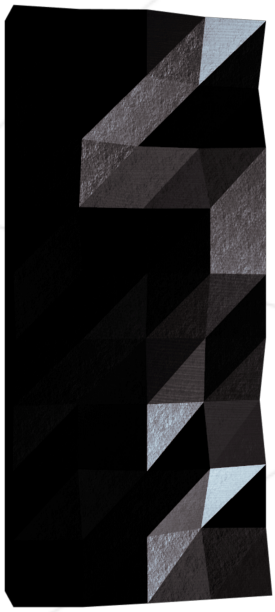

Projet UNIX

ft_shield

Summary: This project will make you code a simple trojan horse.

Version: 4

Contents

Chapter I

Foreword

Chapter II

Introduction

A trojan horse is a special kind of malicious software, or malware, often mistaken with virus or parasites. The trojan horse looks like a legit software, but it contains a malicious parasite. The goal of the trojan horse is to get this parasite inside the computer and install it unbeknownst to the user.

The contained program is called "`payload`". It can be any kind of parasite: `virus`, `keylogger`,`spyware`... This parasite is the only element that will execute the actions inside the victim's computer. The trojan horse is just the vehicle that "puts the fox in charge of the henhouse". It's not malicious itself because it can't execute any action, other than installing the real parasite.

In common language, the parasite is often called "trojan horse" by metonymy. This confusion is often fueled by antivirus publishers.They use "trojan" as a generic name to describe different kinds of malware that are all but trojan horses.

Chapter III

Objectives

This project follows `Matt_daemon` as well as `Dr_quine`, which aim to make you create a simple trojan horse!

This trojan will not be complicated in its core, but with a few bonuses and a little drive, you will quickly understand how you can improve your program.

Chapter IV

Mandatory part

`ft_shield` is a binary of your own design that will only launch with the `root` rights.

This program will be harmless at first (hehe...). When launching `ft_shield`, it will only show your login on the standard output. **NEVERTHELESS**, in the background, it will have made so much more:

- `ft_shield` will create another program also called `ft_shield`, that will be located in the folder containing all the targeted OS binaries and will execute.
- This newly created program will have to be executed when the targeted machine is turned on. You will decide of the launching method. Be creative.
- This program will be launched as a background process just like a daemon.

This is trojan, for you... But what's the use, you'll ask me? Let me tell you:

- You should be able to launch only one daemon instance.
- The daemon will have to plug into 4242 port.
- It has to offer a 3 clients connection simultaneously.
- When a client connects to the daemon, a password will be required. We ask for a secured password (a password in clear in the code is a 0, period).
- When connection is established with a client, the daemon must offer to launch a shell with root rights. You will not be able to access the shell with the `shell` command.

`Warning! No error message must appear during the creation or the use of the binary on the targeted machine!!!!!!`

Chapter V

Use exemple

Here is a possible use exemple:

```
# ls -al /bin/ft_shield
ls: cannot access /bin/ft_shield: No such file or directory
# service --status-all | grep ft_shield
# ./ft_shield
wandre
# ls -al /bin/ft_shield
-rwxr-xr-x 1 root root 12384 Apr 4 14:02 /bin/ft_shield
# service --status-all | grep ft_shield
    [ + ]  ft_shield
# service ft_shield status
. ft_shield.service - (null)
    Loaded: loaded (/etc/init.d/ft_shield)
    Active: active (running) since Mon 2016-04-04 14:08:18 CEST; 1s ago
    Process: 10986 ExecStart=/etc/init.d/ft_shield start (code=exited, status=0/SUCCESS)
    CGroup: /system.slice/ft_shield.service
            -> 10988 /bin/ft_shield
# su wandre
$ nc localhost 4242
Keycode: 42
Keycode: 4242
$>
$> ?
?    show help
shell   Spawn remote shell on 4242
$> shell
Spawning shell on port 4242
$ nc localhost 4242
id
uid=0(root) gid=0(root) groups=0(root)
exit
$ nc localhost 4242
Keycode: ^C
$ su
# netstat -tulnp | grep 4242
tcp       0      0 0.0.0.0:4242          0.0.0.0:*               LISTEN      10988/ft_shield
#
```

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.

Here are some bonus ideas:

- Quantify the I/O data (logically requires a client).
- Add functions to our little program (log the user's actions for instance).
- Use a method that will really disguise your trojan.
- Use a `packing` kind of method right on the trojan in order to make the binary as light as possible.
- Optimize so the executable is hard to detect.

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.
- You will have to code in `C` (any version) or in `ASM` and turn in a Makefile (respecting the usual rules).
- You must manage errors a reasonable way. Your program should never quit unexpectedly (segmentation fault, etc).
- The choice of the OS is yours.
- You will have to be in a VM for the evaluation. For your information, the grading scale was built with a stable 64 bits Debian 7.0.
- You can use anything you will need except libraries that will do the dirty work for you. This would be considered cheating.
- You can post your questions on the forum, Jabber, IRC, Slack...