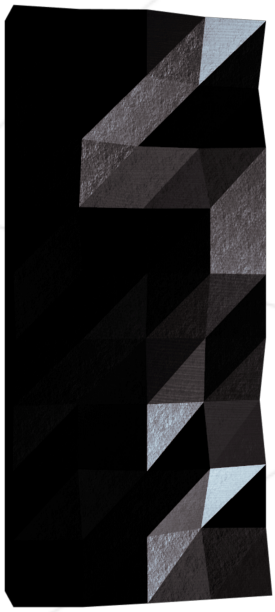

ft_malcolm

An introduction to Man in the Middle attacks

Summary: This is the first project of a network security branch created by maabou-h.

Contents

Chapter I

Foreword

Yes, no, maybe
I don't know
Can you repeat the question?
You're not the boss of me now
You're not the boss of me now
You're not the boss of me now, and you're not so big
You're not the boss of me now
You're not the boss of me now
You're not the boss of me now, and you're not so big
You're not the boss of me now, and you're not so big Life is unfair, so i just stare at the stain on the wall where
The tv'd been, but ever since we've moved in it's been empty
Why i, why i'm in this room
There is no point explaining
You're not the boss of me now, and you're not so big
You're not the boss of me now
You're not the boss of me now
You're not the boss of me now, and you're not so big
Malcolm in the middle, and i confess
I like this mess i've made so far
Grade on a curve and you'll observe
I'm right below the horizon
Yes,...

This subject has nothing to do with Malcolm in the middle, but with Man in the middle!

Chapter II

Introduction

In this first project of network security, you will implement the Address Resolution Protocol spoofing/poisoning method, which is one of the most basic Man In The Middle attacks using a vulnerability present in the ARP protocol.

Chapter III

Goals

In this project aimed at introducing you to network security, you will discover in details what is the Address Resolution Protocol, that you may have seen during your days of piscine as ARP, and how it works. You will find that this implementation has several vulnerabilities and while protections and alternatives to arp might exist, it remains widely used and is unsafe and unprotected in most cases. The ARP protocol lies within the second layer of the complex but wonderful OSI model.

`You should really start by reading the RFC 826 and 7042`

Chapter IV

General instructions

- This project will be corrected by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules
- You must use C and submit a Makefile
- You are allowed to use one global variable within your project.
- Your Makefile must compile the project and must contain the usual rules. It must recompile and re-link the program only if necessary.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Within the mandatory part, you are allowed to use the following functions:
  - sendto, recvfrom
  - socket, setsockopt
  - getuid, close, signal
  - inet_addr, gethostbyname, getifaddrs, freeifaddrs
  - htons, ntohs
  - printf and its family.
  - Your libft functions.
  - You are allowed to use other functions to complete the bonus part as long as their use is justified during your defense. Be smart.

Chapter V

Mandatory part

Your program must take the following four arguments:
source ip
source mac address
target ip
target mac address
Those parameters must always be given in that order.

Example usage:

```
foo@debian:~/projetsecu42_1/sources$ sudo ./ft_malcolm 10.12.255.255 ff:bb:ff:ff:ee:ff 10.12.10.22 10:
    dd:b1:**:**:**
Found available interface: eth0
An ARP request has been broadcast.
   mac address of request: 10:dd:**:**:**:**
   IP address of request: 10:12:12:07
Now sending an ARP reply to the target address with spoofed source, please wait...
Sent an ARP reply packet, you may now check the arp table on the target.
Exiting program...
```

In that example, the program sends an ARP Reply packet to the target (10.12.10.22) containing information on the source (10.12.255.255 and it's associated (spoofed) mac address) right after detecting that an ARP request has been sent over the network.

You must only manage a simple IPv4 address for the mandatory part.
If the arguments provided do not follow that rule or are invalid, your program should exit and output some information.
How precise the information is will be up to you, but you are encouraged to provide details on the errors.

Example error handling:

```
foo@debian:~/projetsecu42_1/sources$ sudo ./ft_malcolm 10.11.11.11 aa:bb:cc:dd:ee:ff 10.11.11.1111 aa:
    bb:cc:dd:ee:ff
ft_malcolm: unknown host or invalid IP address: (10.11.11.1111).

foo@debian:~/projetsecu42_1/sources$ sudo ./ft_malcolm 10.11.11.11 aa:bb:cc:dd:ee:ff 10.11.11.11 aaa:
    bb:cc:dd:ee:ff
ft_malcolm: invalid mac address: (aaa:bb:cc:dd:ee:ff)
```

When started, your program will have to wait for an ARP request sent on the broadcast before sending a single ARP reply to the target and exit.
If everything went well, the arp table on the target should contain the associated ip and mac you provided as source.

You are only allowed to spoof IPs that belong to you, such as your VM. Spoofing other IPs might result in problems and/or sanctions.

Your program should be able to exit when the user inputs Ctrl+C

If you haven't noticed yet, the socket calls in C require low-level privileges, so your program must only run as root/sudo user.

Chapter VI

Bonus part

We will look at your bonuses if and only if your mandatory part isEXCELLENT. This means that your must complete the mandatory part,beginning to end, and your error management must be flawless, even incases of twisted or bad usage. If that's not the case, your bonuseswill be totally IGNORED.

Find below a few ideas of interesting bonuses:

• IPv6 management. • Decimal notation for IPv4 addresses • Verbose mode to print packet information • Any other bonuses that provide extra functionalities to your program

Chapter VII

Turn-in and peer-evaluation

• Submit your work on your GiT repository as usual. Only the work on your repository will be graded.

• You have to be in a VM with a Linux kernel > 3.14. Note that grading was designed on a Debian 7.0 stable.