Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2013

Paulo José Augusto Lopes

Metodologias para Monitorização Integrada de Redes

Methodologies for Integrated Network Monitoring

Paulo José Augusto Lopes

Metodologias para Monitorização Integrada de Redes

Methodologies for Integrated Network Monitoring

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Prof. Dr. Paulo Salvador e do Prof. Dr. António Nogueira, Professores Auxiliares do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Rui Luis Andrade Aguiar
Professor Associado com Agragação da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Paulo Jorge Salvador Serra Ferreira
Professor Auxiliar da Universidade de Aveiro (Orientador)

Prof. Doutor António Manuel Duarte Nogueira
Professor Auxiliar da Universidade de Aveiro (Co-Orientador)

Doutor Eduardo Oliveira Estanqueiro Rocha
Investigador Associado da Leipzig University of Applied Sciences

agradecimentos / acknowledgements

Agradeço à minha família, em especial, aos meus pais, irmã e avó pela forma como me apoiaram ao longo de todo o meu percurso académico. Foram anos recheados de altos e baixos e, tal como estiveram presentes para me congratular pelos sucessos alcançados, também pude contar sempre com eles nos momentos de maiores dificuldades. Por isso e muito mais, tenho a certeza que sem eles nada disto teria sido possível.

Agradeço também aos meus amigos mais próximos por terem estado sempre ao meu lado. Sem dúvida que o apoio deles foi fundamental para me ajudar a alcançar este objectivo.

Por último, um agradecimento especial ao Professor Paulo Salvador e ao Professor António Nogueira, meus orientadores, pelo apoio prestado e conhecimento transmitido durante a realização desta dissertação, bem como pelo tempo dispendido em prol deste projecto.

Palavras-chave SNMP, MAC Spoofing, IP Spoofing, Descoberta da Rede.

Resumo

Dada a importância que as redes assumem nos dias de hoje, é fundamental garantir comunicações sem falhas e, nesta área, a gestão de redes tem tido um papel crucial através da utilização de diversas ferramentas de monitorização. A camada de Dados e a camada de Rede do modelo OSI usam, respectivamente, endereços MAC e endereços IP para proporcionar a comunicação entre os diferentes dispositivos de rede. Uma vez que este é um modelo bastante usado, é frequentemente explorado para actividades maliciosas. Ataques de IP spoofing e MAC spoofing são fonte de várias ameaças à segurança das redes, pelo que prevenir estes ataques é essencial para se obter uma rede protegida e de confiança.

Esta dissertação apresenta alguns mecanismos eficientes de apoio à administração de rede através da realização de tarefas de monitorização específicas baseadas no uso do protocolo SNMP, que é suportado por grande parte do equipamento de rede existente no mercado. O SNMP permite aceder remotamente aos dispositivos de rede e obter informação contida nas suas MIBs. Numa primeira etapa, foi proposto um algoritmo de descoberta da topologia da rede que permite identificar os dispositivos presentes nesta, tal como obter informação útil da rede através da seleção e manipulação da informação da MIB; de seguida, e seguindo o mesmo princípio, foi apresentado um algoritmo para detetar ataques de MAC spoofing e IP spoofing. Foram realizados vários testes de avaliação de desempenho e os resultados obtidos provaram que as metodologias desenvolvidas fornecem um conjunto completo de ferramentas de monitorização de redes capaz de encontrar qualquer dispositivo que suporte SNMP e de rapidamente e eficientemente detetar e bloquear ataques de MAC spoofing e IP spoofing.

Keywords SNMP, MAC Spoofing, IP Spoofing, Network Discovery.

Abstract

Due to the importance of communication networks on current days, it is essential to ensure seamless communications. Network management has a crucial role in this area, through the use of many monitoring tools. The Data Link and Network layers of the OSI model use, respectively, MAC addresses and IP addresses to provide communication between the different network devices. Since this is a widely used model, it is frequently explored for various malicious activities. IP spoofing and MAC spoofing attacks are the origin of many security threats, so preventing them is essential to obtain a protected and trustful network.

This dissertation presents some efficient mechanisms to support network administration by performing specific monitoring tasks, based on the use of SNMP protocol, which is supported by most of the existing network equipment. SNMP allows to remotely access network devices and retrieve information contained in their MIBs. On a first stage, this Thesis proposes a network discovery algorithm that allows identifying the devices present on the network as well as obtaining useful network information by selecting and manipulating the MIB information; then, following the same principle, the Thesis presentes an algorithm to detect both IP and MAC spoofing attacks. Many performance evaluation tests were conducted and the obtained results proved that the developed methodologies provide a complete set of network monitoring tools that are able to find any network device that supports SNMP and quickly and efficiently detect and block MAC and IP spoofing attacks.

Contents

List of Figures

List of Tables

Acronyms

ACL Access Control List

AP Access Point

API Application Programming Interface

ARP Address Resolution Protocol

CDP Cisco Discovery Protocol

CLI Command-Line Interface

CPU Central Processing Unit

DHCP Dynamic Host Configuration Protocol

DTF Destination Traffic Fingerprint

Emerald Event Monitoring Enabling Responses to Anomalous Live Disturbances

FCAPS Faults, Configuration, Accounting, Performance and Security

GUI Graphical User Interface

HCF Hop-Count Filtering

IDS Intrusion Detection System

IOS Internetwork Operating System

IP Internet Protocol

IP2HC IP-to-Hop-Count

IPM Internetwork Performance Monitor

IPS Intrusion Prevention System

IT Information Technology

LAN Local Area Network

LMS LAN Management Solution

MAC Media Access Control

MIB Management Information Base

NAC Network Access Controller

NCM Network Configuration Manager

NFR Network Flight Recorder

NIC Network Interface Controller

NIDS Network Intrusion Detection System

NMS Network Monitoring System

NPM Network Performance Monitor

OID Object Identifier

OSI Open Systems Interconnection

OUI Organizationally Unique Identifier

QoS Quality of Service

RAM Random Access Memory

RIP Routing Information Protocol

RME Resource Manager Essentials

RWAN Router WAN

SNMP Simple Network Management Protocol

SQL Structured Query Language

SSH Secure Shell

SSL Secure Sockets Layer

TCP Transmission Control Protocol

Telnet Telecommunications Network

TFTP Trivial File Transfer Protocol

TTL Time To Live

UDP User Datagram Protocol

VASE Virtual Anti-Spoofing Edge

VLAN Virtual Local Area Network

VPN Virtual Private Network

Chapter 1

Introduction

1.1 Motivation

Today, networks have a fundamental role in our lives, being used for business, communication, data exchange, entertainment, and so on. Due to this increasing importance, networks have been improved in order to become more resilient, secure and able to cope with the appearance of new technologies and applications. The seven layer OSI model was adopted by most of the systems to provide communication between devices. Layer 2, called Data Link layer, uses a physical MAC address to provide communication between the different devices in a local network. This address is a serial number that uniquely identifies the device. Layer 3, called Network Layer, is responsible for packet routing functions, using the IP protocol to deliver packets from source to destination based on their IP addresses.

Due to the importance that networks acquired, it is essential to properly manage all its constituent devices and connections. The network administrator must be able to monitor the entire network and its complete activity. Network monitoring can be described as a set of tasks that constantly monitors the performance and usage of a network and notifies the network administrator whenever a certain relevant event happens. It became an important part of of the network running process, by providing the necessary information to keep all operations under control. Thus, many network monitoring tools have been developed in order to perform different monitoring tasks. These tasks go from simple information about the network topology and devices, detection of network failures, control over the exchanged data or detection of malicious intruders.

Monitoring a network is especially important in those cases where communication networks are fundamental to run a business. In these situations, a network failure could be equivalent to lost of revenues, so it needs to be fixed as soon as possible. Knowing each device present in the network, knowing how they are connected and their performance are just some examples of monitoring tasks that could save a lot of time on the detection of network failures and, therefore, reduce the losts that are caused by these outages. Despite this situation, network monitoring tools are always useful to support networks administration, so it is not surprising that the network management area has been increasingly explored and developed.

To have a trustful network, it is also necessary to take security issues into consideration. A lot of techniques using different approaches have been created to get unauthorized access to networks and perform different malicious activities. These network attacks take advantage

of network failures to affect their targets on many ways. The objective could be intercept secret information that only legitimate users can have access, overload a server or a network connection, among many others. However, having in consideration the method used by the intruder to perform the attack, it is possible to develop techniques that are able to detect and block them. For this reason, the creation of monitoring tools directed to network security has been intensively explored.

1.2 Objectives

As previously said, regardless of the objectives for which a network is deployed, monitoring tools have become fundamental. This work is focused on the development of a set of methodologies for network monitoring: it will be proposed a method for network discovery and two different methodologies for the detection of network attacks. In the first case, the developed methodology will find and distinguish all devices present on the network and retrieve information from them. The retrieved information is related not only to the device but also to the network itself, by consulting routing tables and ARP tables in case of Layer 3 devices and forwarding tables from Layer 2 devices. In terms of the detection of network attacks, the developed methodologies are specifically focused on spoofing attacks. In this type of attacks, the intruder tries to get access to a network for which he doesn't have authorization by impersonating a legitimate user. So, basically the attacker fakes his own access data in order to be able to perform malicious activities over the network. Depending on the faked information, a particular case of spoofing attack will be triggered. If the intruder changes the MAC address of the host he is using to access the network in order to match the one of an authenticated client, we are facing a MAC spoofing attack. Another particular case is the IP spoofing attack which, following the same idea, consists of the configuration of the IP address of the attacker's host in order to be the same as the client IP address. Thus, both methodologies developed for the detection of network attacks will treat the problem of spoofing attacks, focusing particularly on MAC spoofing and IP spoofing attacks.

The proposed approaches to perform network discovery and detect the previously mentioned network attacks will be based on the SNMP protocol. SNMP is used to remotely manage network devices by using data stored on their MIB and is currently supported by most of the network devices. A MIB is a virtual database with information about the network and the device itself. This information is hierarchically organized and each object is identified by the OID. It is possible to detect and block MAC and IP spoofing attacks, as well as perform network discovery, simply by retrieving and managing the information contained on the MIB of each network device.

In summary, the main objective of this project is the development of open-source methodologies that can be integrated on operating networks in order to support network administration. In particular, the work is focused on the development of three different monitoring tools dedicated to specific tasks. For each one, an algorithm exclusively based on SNMP will be presented, together with a possible implementation.

1.3 Contributions

SNMP is becoming globally accepted by all network equipment manufacturers, while the functionalities that this protocol can provide in terms of network management and moni-

toring are still being proposed and developed. In this context, the methodologies that were developed in this dissertation, and were described in the previous section, can be considered somehow innovative. After an intensive research, it was observed that there are already some proposed solutions for network discovery using SNMP, but there isn't any solution that uses this protocol to detect spoofing attacks or any other type of network attack. Thus, given the inovative characteristic of the developed work, a paper titled "Algorithms for Network Discovery and Detection of MAC and IP Spoofing Security Attacks" was written, proposing methodologies and solutions for network discovery and detection of MAC spoofing and IP spoofing attacks based on the SNMP protocol. This paper was accepted for publication at the 5th International Conference on Emerging Network Intelligence (EMERGING 2013), which will take place from September 29th to October 3rd 2013 at OPorto, Portugal.

1.4 Dissertation Structure

This dissertation is structured as follows:

- Chapter two introduces the network monitoring thematic. It will describe some of the protocols that are used for network monitoring and to remotely access network devices; then, some of the most used Network Monitoring Systems (NMS) will also be presented and, at last, it will define and describe the MIB of a network device, whether it was developed by Cisco Systems or any other manufacturer;
- Chapter three is focused on spoofing attacks. It will make a description of this type of attacks, in particular, MAC spoofing and IP spoofing attacks; it will discuss some detection approaches and methodologies and present some of the available Intrusion Detection Systems (IDS);
- In chapter four the developed network discovery methodology will be presented. First, the algorithm will be described and, then some considerations will be made about; finally, a possible implementation for the algorithm will be proposed;
- Chapter five is focused on the two developed methodologies for the detection of spoofing attacks. This chapter is divided into two main sections: the first section will present the algorithm for MAC spoofing detection and the second will present the IP spoofing detection algorithm. Each section will start by describing the algorithm that was proposed for the attack detection and, then, the algorithm for attack blocking. At last, practical implementations for both algorithms will be suggested;
- Chapter six will describe all equipment, virtual and real, used for the creation of the testing scenarios, as well as the software that was used on the simulations conducted to validate the developed algorithms;
- In chapter seven, the network scenario created to test the developed methodologies will be explained; then, all test results obtained from the network simulations and from the execution of the implemented algorithms will be presented. An analysis of these results will also be made;
- Finally, chapter eight will present some possible future enhancements to the developed algorithms, as well as other monitoring methodologies that could be created using SNMP the protocol; finally, the global conclusions about the developed work will be presented.

Chapter 2

Network Monitoring

2.1 Introduction

The current growth on networks complexity demands efficient and secure methods to monitor and manage communication networks. Those methods will allow an easy monitoring of the network performance and the detection of any failures that could arise, keeping basically the network under control. Nowadays, it is essential that any network administrator uses some software or tool that automatically performs these monitoring tasks in order to facilitate his work and, essentially, to turn it more efficient. Depending on the context where the network is deployed, different network characteristics need to be analyzed. For this reason, many network monitoring systems have been created to perform specific monitoring tasks. A huge variety of these systems is currently available, each one using different approaches and techniques.

The first section of this chapter intends to discuss some of the protocols used to support remote monitoring applications and provide access to the information contained in network devices. Then, some of the most used and popular NMS for network administration will be presented. This will allow to create a general idea of the monitoring tools provided by these systems, as well as the methods that were adopted to develop them. Finally, the concept of MIB will be explained and some examples of its contents will be presented. MIBs are used together with the SNMP protocol to manage network devices. As it will be seen in the next section, SNMP will be the protocol used to support the network monitoring methodologies developed on this dissertation and, for this reason, it is essential to understand it, as well as its relation with the MIB of the devices.

2.2 Network Monitoring Protocols

Many network protocols have been created in order to provide access to remote hosts from a local computer, inside a LAN or over the Internet. These hosts can either be simple end hosts or network devices and these protocols work, normally, on a client/server principle. By getting access to these remote devices, it is possible to manage them and perform a set of operations by executing specific commands. Some of the operations available allow consulting and obtaining useful information that can be used to develop network monitoring tools. So, the choice of the protocol that is used to remotely access the network devices is fundamental because all the developed work throughout the dissertation will be based on this protocol.

The most popular and more commonly used remote access protocols are Telnet, SSH and

SNMP. The next paragraphs will make a description of each of these protocols, as well as an analysis in order to evaluate which protocol would better fit the dissertation's context. Another commonly used network protocol is CDP, which is a protocol developed by Cisco to perform monitoring tasks over Cisco devices. So, this protocol will also be described later in this section.

2.2.1 Telnet

Telnet is a network protocol used to connect remote machines in the same LAN or over the Internet. It was launched in 1969 and it is known as one of the earliest network protocols [2]. Back on time, computers were synonymous of large mainframes developed to perform different tasks, if a user needed to use different machines, that would lead to many wasted hours of walking to access to the terminal of each machine located in different places and Telnet was initially developed to overcome this problem. It is based on the concept of a connection-oriented session between a client and a server during a relatively large period of time, running over TCP. TCP connection is done to log into a remote machine and the local machine connects to an open port using the IP address or domain name of the server [3, 4]. This allows a machine to have multiple simultaneous sessions by identifying it with the IP address and port number of the client. Once connected, a CLI appears and UNIX based commands must be executed to interact and manage the remote machine. Telnet is mostly used to perform remote management and also to setup and configure network devices like switches, routers or access points. The great advantages of the Telnet functionality is that most operating systems support this tool, besides the fact that most services are accessible via a Telnet connection. The main problems are related to security issues. By default, it doesn't support encryption and most of implementations don't even have authentication, which is a great problem since passwords and other secret information is exchanged between devices and anybody who intercepts these packets can have access to important data. Due to this lack of security, Telnet has been discontinued and replaced by more secure tools.

2.2.2 SSH

One of these tools is SSH. SSH was first published in July 1995 and it is another network management protocol developed to provide remote access primarily on UNIX and Linux environments [5]. Like Telnet, it intends to log into a remote machine over any network to execute commands and transfer files from one device to another. However, SSH uses cryptographic algorithms to authenticate both client and server and provides encryption to all transferred data. In this way, it prevents attackers from accessing any secret information contained in data packets by protecting its integrity, being the most secure tool to access servers over insecure channels. There are several other features provided by SSH, like TCP/IP ports arbitrarily defined, encryption used to protect against spoofed packets and RSA authentication on client and server to prevent network attacks [6]. Regarding the performance of the SSH protocol, we can say that it has a startup time on the order of a second and a transfer rate dependent on the encryption algorithm but directly proportional to the speed of the device. Compared with Telnet, SSH can achieve substantially faster transfer rates on long-distance connections due to compression of transmitted data. For this reason, but mainly for the strong authentication and secure communications of SSH, this protocol is nowadays the most used to access remote devices.

In the context of this dissertation, SSH also allows the execution of commands directly from the local machine without having to actually logging into the remote device. So, it would be possible to use SSH for the creation of the monitoring tools. However, it has the disadvantage that the necessary information for the development of these methodologies is not easily accessible as it is, for example, using SNMP. So, the creation of algorithms is difficult using this protocol. For this reason, SSH was not the chosen tool to support this project.

2.2.3 SNMP

Other used network management tool is SNMP. The first version of SNMP was launched in late 1980s and it covers not only the protocol itself but also the MIB objects. This protocol allows a client or manager to poll network devices (agents) running on a network for specific information [7]. This information is contained in the MIB, a text file hierarchically organized with information about the device and the network. SNMP uses specific commands to access and manage this information and the separation between protocol and management information reduces significantly its complexity [8]. Table 2.1 presents some of the most used SNMP commands that are used to remotely manage the information contained on the device MIB.

Table 2.1: SNMP commands

SNMP Command	Description
snmpget	It uses the SNMP GET request to query for information on a network entity.
snmpgetnext	It uses the SNMP GETNEXT request to query for information on a network entity. For each OID argument, the variable that is lexicographically "next" in the remote entity's MIB is returned.
snmpwalk	It uses the SNMP GETNEXT requests to query a network entity for a tree of information.
snmpbulkwalk	It uses the SNMP GETBULK requests to query a network entity efficiently for a tree of information.
snmptable	It retrieves an SNMP table and display it in tabular form using repeatedly SNMP GETNEXT and GETBULK requests to query for information on a network entity.
snmpset	It uses the SNMP SET request to set information on a network entity.
snmptrapd	It receives and logs SNMP TRAP and INFORM messages.

Usually a device MIB contains a great variety of information. Figures 2.1 and 2.2 show two examples of the returned information after an *snmpwalk* command is executed and how this information is organized. Fig. 2.1 represents the information related with the system that the device is using and 2.2 corresponds to the information returned from the device MIB that contains its ARP table data. As can be seen, only part of the returned information is useful and, therefore, it is necessary a certain knowledge about the protocol itself and the device MIB in order to be able to exclusively select important data.

```
root@paulo-laptop:~#
root@paulo-laptop:~#
root@paulo-laptop:~#
root@paulo-laptop:~# snmpwalk -v2c -c myrouter 10.1.1.1 sysDescr
SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS Software, 3600 Software (C3640-IS-M), Version 12.4(8), RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Mon 15-May-06 16:27 by prod_rel_team
root@paulo-laptop:~#
root@paulo-laptop:~#
root@paulo-laptop:~#
```

Figure 2.1: SNMP command example - System information

```
root@paulo-laptop:~#
root@paulo-laptop:~#
root@paulo-laptop:~#
root@paulo-laptop:~# snmpwalk -v2c -c myrouter 10.1.1.1 at
RFC1213-MIB::atIfIndex.2.1.13.1.1.1 = INTEGER: 2
RFC1213-MIB::atIfIndex.2.1.13.1.1.4 = INTEGER: 2
RFC1213-MIB::atIfIndex.3.1.11.1.1.1 = INTEGER: 3
RFC1213-MIB::atIfIndex.3.1.11.1.1.2 = INTEGER: 3
RFC1213-MIB::atIfIndex.4.1.12.1.1.1 = INTEGER: 4
RFC1213-MIB::atIfIndex.4.1.12.1.1.3 = INTEGER: 4
RFC1213-MIB::atIfIndex.25.1.10.1.1.1 = INTEGER: 25
RFC1213-MIB::atIfIndex.25.1.10.1.1.2 = INTEGER: 25
RFC1213-MIB::atIfIndex.25.1.10.1.1.5 = INTEGER: 25
RFC1213-MIB::atIfIndex.26.1.10.2.2.1 = INTEGER: 26
RFC1213-MIB::atIfIndex.26.1.10.2.2.2 = INTEGER: 26
RFC1213-MIB::atIfIndex.26.1.10.2.2.5 = INTEGER: 26
RFC1213-MIB::atIfIndex.27.1.10.3.3.1 = INTEGER: 27
RFC1213-MIB::atPhysAddress.2.1.13.1.1.1 = Hex-STRING: CC 04 10 51 00 10
RFC1213-MIB::atPhysAddress.2.1.13.1.1.4 = Hex-STRING: CC 03 10 51 00 10
RFC1213-MIB::atPhysAddress.3.1.11.1.1.1 = Hex-STRING: CC 04 10 51 00 11
RFC1213-MIB::atPhysAddress.3.1.11.1.1.2 = Hex-STRING: CC 05 10 51 00 11
RFC1213-MIB::atPhysAddress.4.1.12.1.1.1 = Hex-STRING: CC 04 10 51 00 12
RFC1213-MIB::atPhysAddress.4.1.12.1.1.3 = Hex-STRING: CC 06 10 51 00 12
RFC1213-MIB::atPhysAddress.25.1.10.1.1.1 = Hex-STRING: CC 04 10 51 00 00
RFC1213-MIB::atPhysAddress.25.1.10.1.1.2 = Hex-STRING: B2 EF F4 DA 17 FC
RFC1213-MIB::atPhysAddress.25.1.10.1.1.5 = Hex-STRING: C2 02 10 42 00 00
RFC1213-MIB::atPhysAddress.26.1.10.2.2.1 = Hex-STRING: CC 04 10 51 00 00
RFC1213-MIB::atPhysAddress.26.1.10.2.2.2 = Hex-STRING: 08 00 27 07 49 3D
RFC1213-MIB::atPhysAddress.26.1.10.2.2.5 = Hex-STRING: C2 02 10 42 00 00
RFC1213-MIB::atPhysAddress.27.1.10.3.3.1 = Hex-STRING: CC 04 10 51 00 00
RFC1213-MIB::atNetAddress.2.1.13.1.1.1 = Network Address: 0D:01:01:01
RFC1213-MIB::atNetAddress.2.1.13.1.1.4 = Network Address: 0D:01:01:04
RFC1213-MIB::atNetAddress.3.1.11.1.1.1 = Network Address: 0B:01:01:01
RFC1213-MIB::atNetAddress.3.1.11.1.1.2 = Network Address: 0B:01:01:02
RFC1213-MIB::atNetAddress.4.1.12.1.1.1 = Network Address: 0C:01:01:01
RFC1213-MIB::atNetAddress.4.1.12.1.1.3 = Network Address: 0C:01:01:03
RFC1213-MIB::atNetAddress.25.1.10.1.1.1 = Network Address: 0A:01:01:01
RFC1213-MIB::atNetAddress.25.1.10.1.1.2 = Network Address: 0A:01:01:02
RFC1213-MIB::atNetAddress.25.1.10.1.1.5 = Network Address: 0A:01:01:05
RFC1213-MIB::atNetAddress.26.1.10.2.2.1 = Network Address: 0A:02:02:01
RFC1213-MIB::atNetAddress.26.1.10.2.2.2 = Network Address: 0A:02:02:02
RFC1213-MIB::atNetAddress.26.1.10.2.2.5 = Network Address: 0A:02:02:05
RFC1213-MIB::atNetAddress.27.1.10.3.3.1 = Network Address: 0A:03:03:01
```

Figure 2.2: SNMP command example - ARP Table information

SNMP is an application protocol encapsulated in UDP and currently it has three versions. The main differences from the first to the second version are mainly the addiction of new protocol operations. On other hand, from versions 2 to 3 the differences are more related with security improvements and remote configuration capabilities [9]. Compared to the previous remote access tools, instead of getting access into a remote machine as it is done in Telnet and SSH and then executing commands to consult information as we were working directly on the

device, SNMP simply sends commands from the local machine to obtain information from the server, without having to log into it. This has the advantage that it is only necessary to execute commands in order to get information from any network device that supports SNMP (nowadays most of the devices actually do).

This last point was actually the main reason for the choice of SNMP as the protocol used on this project to remotely access the different devices present in the network and to support the creation of network monitoring methodologies. The ease of use and the simplicity of this protocol allows an efficient development of algorithms that will automatically send SNMP commands to retrieve information contained on each device MIB and manage this data in order to perform network discovery and to detect IP spoofing and MAC spoofing attacks.

2.2.4 CDP

Unlike previous network protocols, which are focused on the remote access and management of hosts, CDP was developed by Cisco to discover Cisco devices on the network. It is a Data Link Layer protocol and must be enabled on each device to become visible to others. Once enabled, the Cisco device sends periodic information from each connected interface to a multicast destination address. In this way, packets are received by all Cisco devices that have CDP enabled and are directly connected to the device. Thus, each device that supports CDP stores the information received from other devices in a table that is updated each time an announcement is received. The table contains different information about all neighbors, like the operating system version, IP addresses or the device host name. After a defined time without receiving information about a certain device, its information is discarded [10].

This protocol could also be a possibility for the development of network monitoring methodologies by using information contained in the table of each device. However, this protocol can only be applied to Cisco equipment, which is a disadvantage since one of the objectives of this dissertation is to develop general methodologies for network monitoring. Furthermore, SNMP is still simpler to use for the development of these methodologies and, for that reason, it is the chosen protocol.

2.3 Network Monitoring Systems (NMS)

A network monitoring system consists of an application that is deployed over a network to constantly perform monitoring tasks such as performance evaluation, network equipment discovery, monitoring the health and status of the devices and notify the network administrator whenever any anomaly is detected [11]. Network monitoring can be considered as a subset of functions associated to network management, which is a concept based on the FCAPS model [12]. FCAPS stands for Fault, Configuration, Accounting, Performance and Security and describes the management categories that define the whole set of network management tasks.

The previous network protocols are employed by network monitoring systems on the development of various monitoring applications. For example, SNMP can be used to gather information from network devices and use this information for the development of monitoring tools. So, this section will present some of these network monitoring systems.

2.3.1 CiscoWorks LMS

CiscoWorks is a management tool developed by Cisco Systems to facilitate the tasks of configuration, administration, monitoring and troubleshooting Cisco networks. The two main packages provided by CiscoWorks are the LAN Management Solution (LMS) and the Router WAN (RWAN) application. Although the packages' names could induce that LMS is directed to switches and RWAN for routers, this is not true. Actually, CiscoWorks LMS is able to look after both switches and routers and the difference to RWAN is mostly related to additional features such as the ACL Manager and IPM [13]. Thus, if the user does't need these two additional applications, CiscoWorks LMS should be enough. There are some other packages associated to CiscoWorks, such as QoS Policy Manager, VPN/Security Management Solution and IP Telephony Environment Monitor, that provide additional network management solutions, but with functionalities that are out of the scope of this dissertation.

So, referring to CiscoWorks LMS, the integration of this software over networks is a solution for the improvement of the accuracy and efficiency of network operations, for a better control over the network with simplified device configurations, faster identification and fixing of network problems and also for more secure networks through the use of access control services and audit of network changes [14]. In terms of features, CiscoWorks LMS has many components, which are associated to the software in order to perform specific management tasks. For example, Campus Manager is an application that draws topology maps and allows to graphically visualize how the network is connected; CiscoView provides a graphical front-panel that displays Cisco devices to simplify the interaction between user and network equipment, while Device Fault Manager provides real-time and detailed detection, analysis and reporting of device faults [15].

An important component associated to CiscoWorks LMS is CiscoWorks RME. This application is responsible for the lifecycle management of Cisco equipment, reducing manual tasks associated to network maintenance. CiscoWorks RME has the following features [16]:

- Inventory management;
- Device configuration management;
- Software image management;
- Change audit services;
- Syslog analysis.

It uses a mix of the CDP and SNMP protocols to request information from the network and it is really useful for large networks with a lot of network equipment. In the context of this dissertation, we will perform the specific task of network discovery, which is related to the inventory management feature provided by CiscoWorks RME. So, the network discovery application that will be presented in a later chapter should be a reliable alternative to the one provided by this software.

2.3.2 SolarWinds

SolarWinds is a company specialized in network management software. Unlike Cisco Systems, which produces network devices and then develop management tools to support

its equipment, SolarWinds only sells applications for network maintenance, monitoring and troubleshooting. This company was founded in 1999 with the goal of creating efficient management tools, but only since 2005 SolarWinds registered a greater growth, being now one of the best producers of network monitoring systems [17].

In terms of products and services, SolarWinds has a wide range of applications, all of them downloadable. For network faults and performance monitoring, a platform called NPM is available. This software intends to be an easy-to-use tool to quickly detect, diagnose and solve performance issues before outages occur. It shows performance statistics in real-time using dynamic network topology maps and it includes dashboards, alerts and reports related to the monitoring tasks. Another feature of NPM is the ability to perform automated network device discovery and to monitor response time, availability and uptime of routers, switches and all SNMP-enabled devices [18]. This functionality is really close to the network discovery application developed under this dissertation because it also uses SNMP to discover network devices. Other features provided by NPM are the hardware health monitoring, network availability and monitoring (also based on the SNMP protocol), which includes switch port mapper, advanced subnet calculator, bandwidth utilization, packet loss, multi-vendor device support or intelligent network alerting. In terms of other applications provided by this software that are based on the SNMP protocol, a custom MIB poller functionality is also available to collect detailed data from the devices MIB and monitor their performance and statistics.

Another important application developed by SolarWinds is Orion NCM, which is a software directed to network configuration and management and can be used individually or integrated with NPM software to display health configuration indicators alongside with performance statistics. Orion NCM itself simplifies the task of managing network configuration files using a web interface, allowing to quickly fix and change any configuration parameters from network devices without having to manually access them through Telnet or SSH [19]. Thus, the most relevant features of this software are its ability to efficiently troubleshoot network issues, manage and remotely control network configurations, schedule automated backups from devices configurations, make the inventory of all network devices present in the network or identify the connection type from the network equipment to each end host (wired or wireless). Orion NCM includes a network discovery funcionality similar to the NPM application, which records the devices information into a database.

SolarWinds also has many other applications available, some of them free, designed to perform more specific monitoring tasks, including a Cisco NetFlow tool that allows configuring Cisco devices via SNMP. So, SolarWinds is a large company that developed a great variety of applications to support network administration, while the methodologies developed in this project will only perform some specific monitoring tasks, constituting a small part of the tools offered by these applications.

2.3.3 Nagios

The last software that will be presented is Nagios. This application was launched in 1999 and works as a network monitoring system destined to organizations that want to identify and solve IT problems before they affect the business process [20]. Unlike CiscoWorks and SolarWinds, Nagios is a free and open-source software written and maintained by a group of developers that constantly creates new plugins to provide new monitoring functionalities and designed to run on Unix operating systems.

Nagios has many projects focused on different tasks. Nagios Core is the monitoring and alerting engine that works as the nuclear application around which many other Nagios projects are developed [21]. So, Nagios Core was designed with an extensible architecture to provide more flexibility and scalability and to allow the addition of different plugins. This application is focused on checking scheduling, execution and processing tasks, as well as event handling and alerting [22]. This means that Nagios Core doesn't perform any specific monitoring task over networks, being the base application that supports other addon projects.

Nagios Plugins are software applications that work as extensions for Nagios Core and are executed by this main program. These plugins are mostly developed by Nagios community members (nearly 2,000 plugins available), even though there are also official plugins developed by the Nagios Plugin team [23]. So, these developed plugins are responsible for the monitoring tools that Nagios offers. Having in account the number of existent plugins, it is normal that there are already several applications to perform a great variety of monitoring tasks in any type of hardware or service.

In general, some of the main features provided by Nagios are its capability to monitor applications, services, operating systems, network protocols, system metrics or infrastructure components, the ability to provide detailed network performance statistics through a web interface, a centralized view of the monitored network, fast detection of infrastructure outages and the ability to provide the correspondent alerts and complete reports with network performance information [24]. Other important advantages of this software are the fact that it is free and open-source, providing full access to the source code, and it has a complete API that allows any user to easily create custom monitoring applications in many programming languages (C, C++, Bash, PHP, Perl, etc.).

In this chapter we described three different network monitoring systems with different characteristics. The previous section explained the reason for the choice of SNMP as the protocol used to perform the network monitoring methodologies developed in this project. All mentioned monitoring systems support this protocol: CiscoWorks and SolarWinds integrated SNMP from the origin, while Nagios as a pluggin that can be added to the core application. The next section will define the concept of MIB, which is closely related to SNMP.

2.4 Management Information Base (MIB)

As was previously said, MIB is a virtual database present in most of network devices. Its content is composed by a great diversity of information related to the device itself and to the network where it is deployed. To turn this information available and accessible for management purposes, the MIB is associated to the SNMP protocol. In this way, when an SNMP command is sent from a local host, it is possible to manipulate the MIB data. This information is organized hierarchically in a tree format, as shown on Figure 2.3, being defined by a unique OID that specifies the object.

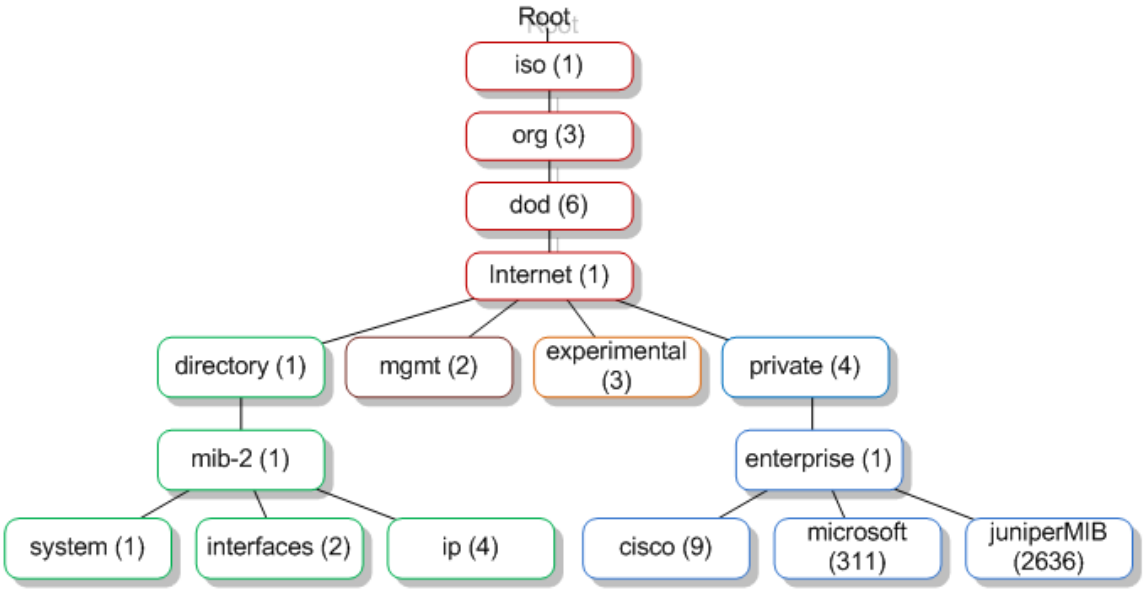


Figure 2.3: MIB tree example. Source: [1]

The OID consists of a sequence of numbers separated by dots. Starting from the top of the tree, each number corresponds to the number of each branch that has to be followed until the desired MIB object is reached. It is also possible to identify the MIB object by the correspondent object name. By executing SNMP commands from a local machine, it is possible to manipulate these MIB objects and, therefore, the information contained in each network device that supports this protocol. Depending on the equipment manufacturer, a device may contain different MIBs with specific objects. So, when using SNMP to perform monitoring tasks, it is essential to know in first place the device manufacturer as well as the specific MIBs that each device contains.

2.4.1 Cisco Equipment

Cisco Systems is the biggest producer of network equipment, providing a great variety of products and services for different market segments. To support all this equipment, Cisco has been developing MIBs, with all necessary information about the devices, that can be used by network management stations. As previously said, SNMP is the protocol that manages the MIB information. Even though each MIB has a great quantity of information, normally only part of it is useful. Table 2.2 presents some of the MIB objects contained in the Cisco IP-FORWARD-MIB. These objects have information about the routing table of Cisco routers, such as destination networks and corresponding network masks, next-hop IP addresses, used interfaces, route types and route metrics.

Table 2.3 presents other MIB objects usually used to perform monitoring tasks. These objects allow retrieve information about IP addresses corresponding to the media-dependent physical addresses and corresponding address types (static or dynamic), MAC addresses and interfaces. The two first objects are contained in the IP-MIB, while the other two are in the RFC1213-MIB from Cisco.

Table 2.2: Some MIB objects from Cisco IP-FOWARD-MIB

MIB Object	OID
ipCidrRouteDest	.1.3.6.1.2.1.4.24.4.1.1
ipCidrRouteMask	.1.3.6.1.2.1.4.24.4.1.2
ipCidrRouteNextHop	.1.3.6.1.2.1.4.24.4.1.4
ipCidrRouteIfIndex	.1.3.6.1.2.1.4.24.4.5
ipCidrRouteType	.1.3.6.1.2.1.4.24.4.1.6
ipCidrRouteMetric1	.1.3.6.1.2.1.4.24.4.1.11

Table 2.3: Some MIB objects from Cisco IP-MIB and RFC1213-MIB

MIB Object	OID
ipNetToMediaNetAddress	.1.3.6.1.2.1.4.22.1.3
ipNetToMediaType	.1.3.6.1.2.1.4.22.1.4
atPhysAddress	.1.3.6.1.2.1.3.1.1.2
atIfIndex	.1.3.6.1.2.1.3.1.1.1

The previously mentioned MIBs are just some of the most commonly objects used on network monitoring. But there is a lot of other MIBs that can contain useful information. For example, the Cisco IF-MIB has information about the interfaces of the device and the BRIDGE-MIB is related to Layer 2 network devices. Cisco has developed MIBs that allow any monitoring task using the SNMP protocol, so we only have to search for the right MIB object of each network device.

2.4.2 Other Equipment

Obviously, Cisco is not the only company that has been developing MIBs to support their network equipment. SNMP became globally accepted and, nowadays, most of network devices support this protocol. Thus, the creation of MIBs was mandatory to manufacturers in order to allow using SNMP for the remote management of their devices. Besides Cisco Systems, some of the major networking equipment companies are [25]:

- Juniper Networks;
- Alcatel-Lucent;
- Huawei Technologies;

These companies are just some examples of network equipment manufacturers that have developed their own MIBs supporting SNMP. Let us start by Juniper Networks, an American company founded in 1996, whose main products are routers, Ethernet switches and security devices. Routers were the first product to be commercialized in 1998 and Juniper's switches were only introduced ten years later, in 2008. Juniper Networks owns also an operating system called Junos, which is run in most of their products. In terms of security equipment, they

produce a line of firewall equipment, security services devices and SSL-based VPN services to provide remote access through regular web browsers on many platforms [26]. To support SNMP, Juniper Networks has created a total of 412 MIBs to cover all their network equipment, making a total of 19,683 MIB objects [27]. One of the main MIBs developed by Juniper, and the one that contains more information, is the JUNIPER-MIB.

Alcatel-Lucent is a French company that provides telecommunications solutions to service providers, enterprises and governments. This company owns Bell Labs, which is one of the largest research and development centres in the communications industry [28]. Alcatel-Lucent is mainly focused on the creation of fixed, mobile and converged networking hardware, Layer 3 technologies working over IP, software and services. Thus, this company is more general than Juniper Networks in terms of produced equipment due to the research lab they hold. In terms of created MIBs that support remote access to the network equipment through SNMP, Alcatel-Lucent developed 215 MIBs with a total of 21,008 MIB objects [29]. The MIB that contains more objects and, consequently, more information is the ADN-MIB.

Huawei Technologies is a Chinese company founded in 1988 that produces networking and telecommunications equipment. In general, Huawei provides operational and consulting services to enterprises in any part of the world and also develops and produces networking equipment for the consumer market. This company have three core business segments: telecom carrier networks, enterprise business and devices manufacturing [30]. The first business segment is centred on the development of network technologies and services, offering mobile infrastructures, broadband access and service provider routers and switches. The second segment consists of services and solutions to support other telecommunication companies and operators and the last one is focused on the production of electronic communication devices like smartphones, modems or wireless terminals, either under its own name or under white-label products that are sold to other companies. To support the SNMP protocol, Huawei Technologies developed 190 MIBs, with a total of 10,781 MIB objects [31]. In this case, HUAWEI-MIB is the one that contains the most important device information.

These three companies represent alternatives to Cisco Systems, even though each one is more targeted to specific business segments. Other companies like D-Link, Netgear or Nokia Siemens Networks are also network equipment manufacturers that could be actual alternatives as well. What all these companies have in common, in the context of this dissertation, is the fact that they have developed their own MIBs to support the remote management of the equipment they produce using the SNMP protocol.

Chapter 3

Spoofing Attacks

3.1 Definition

Dishonest people have always existed, people that try to take advantage of systems for their own benefit. This happens everywhere and the business world is probably the most relevant example of this practice, where system failures are massively exploited to generate huge profits. The virtual world is not an exception, being a field for the development of several malicious activities. Networks had a great development in the last decades, reaching a complexity and robustness level that conducted to their wide usage. Internet is the best example of the importance that networks have acquired. Some typical applications of networks are data exchange, communication, entertainment and business. Due to the variety of usage profiles and information that is exchanged between devices, it is normal that malicious people have developed ways to bypass network security in order to obtain secret information or simply damage networks.

In general, the act of inducing damage on a network is called network attack. There are different types of network attacks, developed to achieve different goals. There are passive attacks, where important information is monitored, and active attacks that intend to corrupt or even destroy important data or the network itself. This dissertation is focused on a specific type of network attack: the spoofing attack. Spoofing consists on the creation of a misleading context in order to lead a victim to make decisions that it will allow gaining access to restricted resources and stealing information [32]. This context can take a variety of forms but it is always based in a scenario in which the attacker pretends to be someone else, usually an authorized client, to have access to certain resources that it wouldn't normally have. Depending on the information the intruder impersonates, there are specific cases of spoofing attacks. Again, for the purpose of this dissertation, two particular cases of spoofing attacks were studied: MAC spoofing attacks and IP spoofing attacks.

The Data Link Layer (Layer 2) of the OSI model uses MAC addresses to identify and provide communication between the different devices of a LAN. The exchanged data inside networks is divide into packets. To know where the packets come from and what are their destinations, the network devices make use of MAC addresses to identify the computers and ensure information is correctly delivered. A MAC address is a permanent address assigned to each network interface of any network device (NIC cards, Wireless adapters, etc) by the hardware manufacturer. These physical addresses are globally unique for each interface and any device connected to a network is identified by the interface MAC address it is using

[33]. Even though it is supposed to be permanent, it is possible to change the assigned MAC address. This is actually the basic principle of MAC spoofing attacks. MAC spoofing consists of changing the MAC address associated to a NIC card, for example, to impersonate another network device or to hide a computer on the network [34, 35]. This will allow bypassing the access control list of servers or routers. The MAC address can also be changed for legitimate reasons, for example to connect to WI-FI hotspots where the internet service provider bind their services to a specific MAC address. However, the focus of this dissertation is the development of measures to detect MAC spoofing attacks, performed for non-legitimate reasons.

Beyond the MAC address, each computer is also identified by an IP address. The Internet Protocol works over the network layer (Layer 3) and routers use these addresses to route data packets across the networks and to provide communication between devices. Thus, in order to identify the origin and destination of information, IP addresses from the corresponding computers are included in the data packets. So, basically, while devices like switches work over Layer 2 and use MAC addresses to forward packets, Layer 3 devices make use of IP addresses to route the packets over different networks. Unlike MAC addresses, IP addresses are not defined and assigned to a device interface during its fabrication. Instead, these addresses are manually configured or assigned by a DHCP server on each device interface according to the network where it is connected to. Thus, changing the IP address associated to a computer interface is a relatively simple process. This is also the main principle of IP spoofing attacks. IP spoofing is defined as the process of configuring a host with the same IP address of a computer with legitimate and authorized access to certain information and resources. As happens in spoofing attacks in general, here the attacker changes his IP address to impersonate a user and gain unauthorized access [36, 37]. This type of attacks is more directed to communication between distant computers because routers are responsible for routing the packets by analyzing the destination address but, generally, they ignore the originating address, which is only used by the destination computer to answer back to the source. The destination host will believe that the messages come from a trustful source and this is actually the essence of IP spoofing attacks.

MAC spoofing and IP spoofing are relatively similar network attacks, with the difference that each one gets access to restricted information using different data to impersonate an authorized user. While to perform a MAC spoofing attack the intruder changes the MAC address of his host in order to match the MAC address of a legitimate client's host, on IP spoofing attacks the attacker uses the IP address to fool the victim. In the next section, different approaches to detect these types of network attacks will be presented .

3.2 Attack Detection Methodologies

The previous section made a description of spoofing attacks, in particular, MAC spoofing and IP spoofing. It explained in what these attacks consist and how they are performed. Due to the appearance of these security threats against networks, it was necessary to develop applications that could detect them. One of the objectives of this dissertation is precisely the creation of methodologies for the detection of these two types of spoofing attacks. Thus, since they were already defined, it is important to know the different approaches that can be taken in order to develop counter-measures against these network attacks. Having this in mind, different solutions can be found to solve this problem. This section will discuss two general

methodologies for the detection of spoofing attacks: from the network point of view, there can be a local and a distributed approach. Then, each approach will be described, as well as how it can be deployed. Besides, some already developed methodologies that use each one of the approaches will also be presented. Taking a look to other projects that have already been developed and focus on the same subject will allow to create the basis for the work that will be presented in later chapters for spoofing attack detection.

3.2.1 Local

The first method that will be presented for the detection of spoofing attacks is the local approach. In general, the local method consists on adding probes in specific places of the network. These probes will perform specific tasks defined by the developed methodology, for example, capture data packets that pass through that point of the network and analyze the information that it contains in order to detect network attacks. So, an attacker can send spoofed packets to the network, but this approach will only detect the spoofing attack on well defined locations. To have a better knowledge of this approach, we will now present some already developed and implemented methodologies.

Referring to the detection of MAC spoofing attacks, a paper that uses this local approach is called "A design of egress NAC using an authentication visa checking mechanism to protect against MAC address spoofing attacks" [38]. An egress NAC is used to authenticate internal users before accessing external networks by protecting and controlling them when browsing the Internet, for example. It is mostly used on WI-FI hotspots, but it can also be used on wired connections using Ethernet ports. MAC spoofing can easily bypass the egress NAC by spoofing the MAC address of an authenticated client and getting access to the network. This paper proposes new egress NAC based on an authentication visa checking mechanism to solve this problem. Normal NACs use IP and MAC addresses to identify the authenticated clients, which can be fooled if an attacker spoofs these addresses. The authentication visa created in this paper uses messages generated by a security agent applet as an additional factor to validate users. So, this methodology can be considered as a local approach to detect spoofing attacks because the new NAC is placed in a specific point of the network to verify the authenticated clients.

Anther developed methodology that uses a local approach, but now for detection of IP spoofing attacks, is presented in a paper named "Defense Against Spoofed IP Traffic Using Hop-Count Filtering" and is based on the fact that even though an attacker can forge any field of the IP header, he cannot fake the number of hops an IP packet takes to reach its destination [39]. Thus, when an intruder spoofs an IP address, he will not be able to manipulate the hop-count for the same value of the victim. Due to the ease that an Internet server has to obtain the hop-count information from the TTL field of the IP header, it is possible to create a map of IP addresses and their correspondent hop-counts in order to detect spoofed IP packets. This filtering technique is called HCF and uses an IP2HC mapping table that will detect and discard spoofed IP packets. This is another example of a local approach for the detection of IP spoofing attacks, where the filtering system is placed near the possible victim of the attack, like a server or another host, in order to avoid that it could damage this specific device.

Next paper, entitled "VASE: Filtering IP spoofing traffic with agility", basically proposes another method to perform IP spoofing filtering with a reduced resource consumption, which will be proportional to the size of the attack [40]. The filtering mechanism is called VASE

and it uses sampling and on-demand filter configuration to detect IP spoofing attacks and, at the same time, reduce unnecessary overhead due to the existence of intermittent attacks.

3.2.2 Distributed

The other approach that could be taken to detect spoofing attacks, or any malicious activity in general, is a distributed one. Unlike the local approach, which analyzes a specific location of the network, the distributed approach is able to analyze data packets in different points or analyze many network devices in order to detect network attacks. This allows to detect the presence of spoofing attacks in any location of the network, protecting the whole network instead of a single device. Let us now mention some projects that were developed using this approach.

Paper "Network Simulation for MAC Spoofing Detection, using DTF Method" [41] proposes a method to detect MAC spoofing attacks. As the title says, this paper proposes a MAC spoofing detection methodology based on a DTF. The general idea of this method is to generate traffic from an end device connected to the network to a set of IP destinations. Each destination will have a constant traffic in time that will be used as a reference fingerprint. For each fingerprint, the IP address and the percentage of traffic from that destination are recorded. The reference fingerprint is compared to the actual fingerprint and the method establishes the Overall Degree of Recognition that will determine if a MAC address is being spoofed or not. Obviously, this method uses a distributed approach for the detection of MAC spoofing attacks because information is obtained from different end hosts, which will allow to detect if any of them is performing an attack.

A paper that uses a distributed approach to deal with IP spoofing attacks is titled "A Trust-based Approach against IP-spoofing Attacks" [42]. This paper proposes a method based on a Bayesian inference model to detect attacks performed by access routers. Most of the detection mechanisms assume that IP spoofed packets are generated only by end hosts, but the truth is that even though they send genuine traffic, access routers can modify the source IP address of the packets before forwarding them. This model evaluates the trustworthiness of the routers based on the number of detected IP spoofed packets through the application of the referred inference model by a judge router. Each access router sends a copy of every packet they forward to this judge router, which computes the trust values for them. This methodology also avoids that IP spoofed packets travel the network to reach the destination by performing the attack detection on the source side, reducing wasted network resources. This methodology can be considered as a distributed approach to detect IP spoofing attacks because it analyzes packets forwarded by all access routers and detects attacks performed at any point of the network, instead of analyzing a specific location.

At last, an approach named "An Effective Method for Defense against IP Spoofing Attack" is based on traceroute and cooperation between trusted adjacent nodes in order to detect and block intruders from external networks [43]. Without entering in detail on this method, it can be immediately seen that this is a distributed approach because different network nodes are analyzed, which means that many network probes are placed over the network to perform the IP spoofing detection.

In summary, the local approach can be considered a passive method on the detection of network attacks. In general, it consists on a packet filtering placed in a specific point of the network that continually analyzes the data packets that pass through it until a spoofed packet

is detected and discarded. On the other hand, the distributed approach has a more active role on the detection of spoofing attacks. This methodology analyzes different locations of the network, seeking for the attacks instead of waiting for them in a specific point. Both approaches have advantages and disadvantages: the local methodology leads to less usage of resources but the distributed approach is more efficient on the detection and blocking of the attacks. For the purpose of this dissertation, the SNMP protocol will be used to perform network discovery and detection of MAC spoofing and IP spoofing attacks. This protocol will retrieve information from the different network devices, which will be used to develop the monitoring methodologies. So, in this method, each network device will work as a probe and the approach that will be presented for the detection of spoofing attacks is a distributed one.

3.3 Intrusion Detection Systems (IDS)

An Intrusion Detection System is an application developed to monitor network traffic and look for malicious activities. These security monitoring systems gather and analyze data from many network locations in order to identify and detect possible system intrusions and misuses. An intrusion is considered as an attack performed from outside the network and, therefore, outside the organization, while a misuse is an intrusion generated from inside [44]. An IDS is focused on the detection of network attacks from both inside and outside and, in some cases, it may also take some actions and block the source of the attacks.

There are different variants of IDS that deal with the detection of suspicious traffic in different ways. These systems can be grouped in two types: Network based IDS and Host based IDS. As the name suggests, while the first one is placed in strategic places within the network to monitor all the devices on the system, the second type is run inside hosts or network equipment to protect only that specific device. Additionally, an IDS can also be categorized according to its detection mechanism: signature based IDSs, anomaly based IDSs and hybrid IDSs. The signature based IDS monitors the network packets and performs the detection based on a comparison of the traffic with specific signatures and attributes of already known threats. The anomaly based IDS establishes a pattern based on the bandwidth, used protocols and ports that are considered normal for each network, monitoring the network traffic and comparing it to this baseline. Finally, hybrid systems combine both IDS mechanisms [45].

Depending on the type of IDS, different tasks will be performed. But in general, an IDS includes the following functionalities:

- Monitoring and analyzing both user and system activities;
- Analyzing system configurations and vulnerabilities;
- Assessing system and file integrity;
- Ability to recognize patterns of typical attacks;
- Analysis of abnormal activity patterns;
- Tracking user policy violations.

Next, we will analyze some free and open source IDS developed to protect networks and, according to the context of this dissertation, allow the detection of spoofing attacks.

3.3.1 Snort

One of the most used free and open-source applications for network protection is Snort, developed by Sourcefire [46]. From the previously mentioned types of IDS, Snort belongs to NIDS. It is considered a lightweight tool, which means that it is a small, powerfull and flexible IDS in order to be easily deployed and a permanent element of the network security infrastructure. In terms of application, Snort is a cross-platform and can be deployed on small TCP/IP networks to detect suspicious network traffic and non-legitimate network attacks without the need for monitoring or administrative maintenance during long periods. Initially, Snort wasn't developed to work as a complete IDS, instead it was developed as a supplement to other IDSs in order to fill some security gaps they could have [47]. However, Snort has been developed and it has increased without leaving the concept of a small application ($\sim$ 75,000 code lines) with a minimal interference in the system and network performance. Comparing with most commercial NIDS, Snort is easier to configure by network administrators and doesn't require a dedicated platform, which leads to a more rapidly implementation and easy to use network security solution. Snort has three execution modes:

- Sniffer mode;
- Packet logger mode;
- Intrusion detection mode.

The sniffer mode allows to read network packets and to display the data contained in the header and body of each packet to the screen. The packet logger mode will basically log the network packets to the disk. Sniffer and packet logger modes have similar functionalities, in which network packets are analysed, with the difference that the first mode writes the data into the screen and the second into the hard drive of the host where it is being executed. These two running modes are suitable on data capture but it is not practical to use this information to detect network intrusions or misuses. For this reason, Snort also includes the intrusion detection mode. In this mode, the user defines a set of rules and the program monitors and analyses the network traffic based on these rules. Then, if some suspicious activity is detected, the system will apply specific counter-measures [48]. According to the defined rules, it is possible to detect a wide variety of intrusions and attacks and this last mode is precisely the one that it is more related with the purpose of the dissertation.

Thus, Snort is more like a rule based than a signature based IDS, providing a simple but efficient way to protect networks against intrusions and attacks. So, the methodologies presented on this dissertation should represent a reliable alternative to this tool.

3.3.2 NFR

One of the most complete IDS in the market is NFR. This software was developed to track attempted break-ins in a system or server from a separated computer, which means that even if the system we are protecting is destroyed or becomes unavailable, NFR always survives. It uses a hybrid based approach, inspecting the OSI model, from Layer 2 up to Layer 7, looking for any suspicious activity [48]. As any other sniffer, NFR provides data analysis and collection by reading the network packets but, in addition, it uses a scripting open-source language called N-Code to perform a complete packet inspection in order to detect network misuses, protocol anomalies and network intrusions [49]. N-Code is a very flexible programming language that

works as a filtering engine that will allow users to configure this IDS in order to sample great portions of network traffic and perform reasonable packet analysis before choosing to record it and evaluate possible network attacks. Thus, this language gives NFR the necessary extensibility for the creation of automated real-time alerts and management tasks.

NFR is an open-source software but it is only free for noncommercial and research purposes. This IDS gives the user the possibility of customizing the software according to their protection needs and authors have also created some more intelligent and programmable tools for network monitoring that allow to detect network attacks in a more efficient way [50]. Unlike Snort, for example, NFR can be applied on large networks and its filters occupy very little memory which, combined with its customizable configuration, turns this IDS into a powerful a very flexible tool for a complete network protection.

3.3.3 Emerald

The last IDS that will be presented is called EMERALD. This software is an example of a distributed IDS using both signature and anomaly-based approaches for the detection and tracking of malicious activities. Like NFR, it was developed for large networks with thousands of users connected and providing real-time responses [51]. EMERALD is also a very scalable tool allowing network surveillance, attack isolation and response monitors that are deployed at various abstraction layers. These monitors combine signature analysis with probabilistic inference to protect systems in real-time. The EMERALD project developed an architecture with well-developed analytical techniques in order to detect different network intrusions and to cast them in frameworks, which are highly reusable and interoperable [52, 48].

This chapter started by defining one type of security threat against networks: spoofing attacks. The network protection tools developed on this project are focused on this type of attack. Then, two general approaches for the detection of malicious activities were presented in order to have a better idea of the possible approaches that can be taken for the development of methodologies for detecting spoofing attacks. Finally, it was presented three open-source and free IDS that use different methods to protect networks against intrusions and misuses. These IDSs provide a complete set of tools to detect attacks and monitor network packets; although the developed work consists only on a small part of all the tasks these tools can perform, it can be a reliable alternative to these solutions on this specific task.

Chapter 4

Developed Network Discovery Algorithm

As previously said, SNMP executes specific commands from a host in order to access the MIB of the different devices present on the network. The necessary information is selected and manipulated to create algorithms that will perform the desired network monitoring tasks. The first one that was developed, and is described in this chapter, is the network discovery algorithm. This algorithm not only finds all Layer 2 and Layer 3 devices, but also gathers useful information about them and the network and records this data into a database so it can be available for monitoring purposes.

To detect spoofing attacks against the network using SNMP, the information contained in each device MIB should also be manipulated. This information will allow detecting the attacks and blocking them. This means that, before the network attack detection, it is necessary to know all devices present in the network because the algorithm needs to know which devices have to be analyzed. So, the network discovery algorithm presented in this chapter can work not only as an individual network monitoring tool but also as a supporting tool for the detection of spoofing attacks, by providing information about the network equipment.

For the purpose of this dissertation, this algorithm is prepared to be deployed only on networks using Cisco network devices. As previously mentioned, different equipment manufacturers have different MIBs and the algorithm was developed for Cisco MIBs. To deploy this method on different equipment, we only have to change the code according to the MIB objects of the corresponding manufacturer and it should work well. Thus, the network discovery algorithm described in this chapter intends to work as a network monitoring tool for Cisco devices and it also provides the necessary information for the correct execution of the spoofing attack detection algorithm presented in the next chapter.

The first section of this chapter will describe the procedure/algorithm that was proposed for network discovery. Then, some limitations of this method and some considerations that have to be taken into account when using this algorithm are presented. Finally, the last section explains one possible implementation for this algorithm and the one that was used for test purposes.

4.1 Algorithm Description

The developed mechanism to discover the whole network is illustrated on Figure 4.1. It starts by accessing an already known router in the network using its IP address. From this router, it retrieves information from its MIB using the SNMP "snmpwalk" command, putting it in an array that can be easily accessed later. First, it retrieves information about the host name and the model of the device. This information is obtained using the MIB objects *hostName* (OID .1.3.6.1.4.1.9.2.1.3) and *sysObjectID* (OID .1.3.6.1.2.1.1.2), respectively. By joining this device data to the IP address of the router that was inserted at the beginning, we have the necessary information to characterize the device.

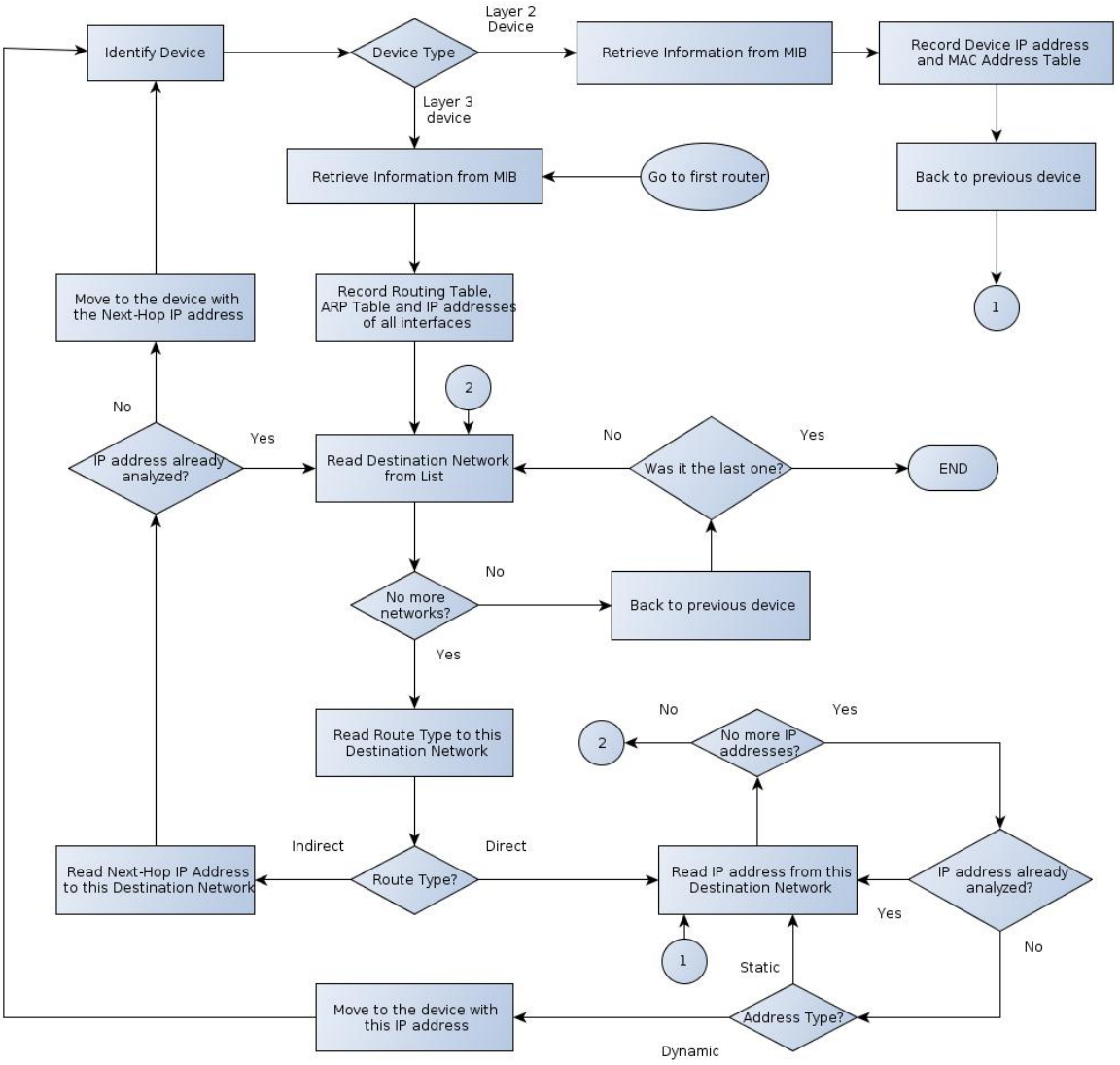


Figure 4.1: Network Discovery Mechanism

Then, it will retrieve information about the routing table of the router. This data is obtained from the MIB objects shown in Table 2.2, which contains information about des-

tination networks, network masks, next-hop IP addresses, used interfaces, route types and route metrics. Actually, the MIB object related to the interface used to reach a certain network only returns the interface index. Thus, the *ifDescr* MIB object (OID .1.3.6.1.2.1.2.2.1.2) can be used to obtain the corresponding interface name. Furthermore, if the route type for each destination network is not direct, it is possible to use the next-hop IP addresses and the corresponding host names to know to which routers is this device connected to. However, since next-hop IP addresses define a packet route to reach a destination network, if the analyzed router is connected to another router that is not defined as next-hop to any destination network, it won't be possible to discover the connection between both.

The information from the MIB objects represented in Table 2.3 is used to get the device ARP table. These objects contain information about the IP addresses corresponding to the media-dependent physical addresses, as well as the associated address types (static or dynamic), MAC addresses and interfaces [53, 54, 55]. Again, the *ifDescr* MIB object is used to get the actual interface name.

A situation that has to be considered in this method is the fact that each router can have several IP addresses associated to each interface. When performing network discovery, each device only needs to be analyzed once; however, since it can have more than one IP address, the algorithm could analyze the same router more than once. This is why the next step is to record all IP addresses associated to each interface in order to assure that the device is analyzed only once. This information is found in the router MIB object *ipAdEntAddr* (OID .1.3.6.1.2.1.4.20.1.1). To associate each of the IP addresses to the right interface, the MIB object *ifDescr* is used , after the corresponding interface indexes have been retrieved using the *ipAdEntIfIndex* (OID .1.3.6.1.2.1.4.20.1.2) object.

The previous steps and all the mentioned MIB objects contain the necessary information about each router and all this data must be retrieved every time a router is analyzed. To have this information available for a posterior use on monitoring tasks, for example, it is recorded on a database. In order to move to the next network device, the destination networks previously retrieved from the router MIB are used. For each destination network, the algorithm must check the route type. If the route type to that network is indirect, the value of the next-hop IP address is read and the algorithm moves to the router with this IP address, following all the previous steps. Since this is the first router, it is possible to move to the next device without checking if it was already analyzed. However, from now on it is necessary to compare the next-hop IP address with the list of IP addresses corresponding to the devices where we have already been. If the route type to a destination network is direct, the IP addresses of all Layer 2 devices present on that network must be read. The IP addresses from these devices can be found on the already retrieved *ipNetToMediaNetAddress* MIB object. Whenever the algorithm finds in the list an IP address corresponding to a network device that was not already analyzed and whose address type is defined as dynamic (because static IP addresses usually belong to the interfaces of the device that it is being analyzed), then the algorithm moves to this new network device. After all Layer 2 devices present on a given network have been analyzed, then the next destination network from the array is read and the route type is checked again. Since this is a recursive algorithm, when there are no more destination networks to reach, we must go back to the previous router that was being analyzed. When the first router that was analyzed is finally reached and there are no more destination networks to move to or Layer 2 devices to analyze in a given network, then it means that all network devices have been discovered. Additionally, it is possible to define a stopping network for cases in which it is not desired that the network discovery algorithm discovers all networks.

Thus, when the list of destination networks from a certain router is being analyzed and the stopping one is detected, the algorithm was defined to ignore this network and proceed to the next destination network from the list.

In the case of Layer 2 devices the task is much simpler. When a router is performing the task of analyzing the list of devices belonging to a directly connected network from the *ipNetToMediaNetAddress* MIB object and a Switch or AP that was not analyzed is found, the algorithm has simply to move there, retrieve the necessary information and then go back to the router and read the next IP address from the list. So, the first thing to do with Layer 2 devices is to record the information that characterizes the device. Similarly to the case of routers, here the recorded information is also the IP address, the device hostname and the device model. Then, information from the device forwarding table will be retrieved. This can be done by retrieving information from the MIB objects represented on Table 4.1.

Table 4.1: Some MIB objects from Cisco BRIDGE-MIB and CISCO-STACK-MIB

MIB Object	OID
dot1dTdbAddress	.1.3.6.1.2.1.17.4.3.1.1
dot1dTpFdbPort	.1.3.6.1.2.1.17.4.3.1.2
dpt1dBasePortIfIndex	.1.3.6.1.2.1.17.1.4.1.2
vlanPortlslOperStatus	.1.3.6.1.4.1.9.5.1.9.3.1.8

The first two objects represent, respectively, the MAC addresses and the corresponding bridge ports from the MAC address table of the device. To convert the bridge port into the actual device interface, the next MIB object from the table should be used. The *dot1dBasePortIfIndex* object allows to get the actual interface index, which can be associated to the interface name using the *ifDescr* (OID .1.3.6.1.2.1.2.2.1.2) object [56]. Then, to obtain the types of the addresses (dynamic or static) and the VLAN associated to each one, the *ipNetToMediaType* and *atIfIndex* objects (Table 2.3) are used . The second one is used as index on the *ifDescr* object to return the corresponding VLAN. Finally, using the *vlanPortlslOperStatus* MIB object, the last one from Table 4.1, it is possible to verify if the bridge port is a Trunk port or an Access port. This process allows retrieving the same information that is obtained when the "show mac-address-table" command is executed in Cisco Layer 2 devices. After the previous steps are executed, the algorithm returns to the router and continues looking for other Layer 2 equipment on that LAN.

We have just described a method that will perform network equipment discovery in any network, using SNMP as the support protocol that allows obtaining the necessary information from each device. When running the algorithm, all Layer 2 and Layer 3 devices present on the network will be discovered and the necessary information that characterizes each device is retrieved. This methodology will also obtain other network information like forwarding tables from Layer 2 devices and routing and ARP tables from Layer 3 equipment. Additionally, it also discovers how routers are connected and retrieves information about each interface. Obviously, more information can be obtained from each device MIB but this algorithm presents a method to consult the most useful information in order to support network monitoring.

4.2 Considerations and Limitations of this Method

The previous section described a method to discover network devices. This is an efficient method that, theoretically, can be deployed in any network. But due to the existence of different network equipment, some situations have to be considered in order to have a general algorithm that can be applied anywhere. This variety of devices may also bring some limitations to this method. In this section, we will describe and explain some of these situations.

Although most of the network devices support the SNMP protocol, there are still some exceptions. For a correct deployment of this method, it will be considered that all Layer 3 devices support SNMP. Otherwise, the network is not correctly discovered and other devices on the network may not be found. However, it is possible to have managed Layer 2 devices (devices that support SNMP) and unmanaged ones. If a network has any unmanaged switch or AP, it won't be detected by the network discovery mechanism but it won't have any other consequence on the discovery of the remaining network. To solve this problem a counter can be created to check how many Layer 2 devices the algorithm analyzes in a certain LAN. This counter will obviously count the number of managed devices. On a managed switch, the *atPhysAddress* MIB object can be used to count the number of Layer 2 devices present in the LAN (even the unmanaged ones). The difference between the two counters corresponds to the number of unmanaged devices. This way, these unmanaged devices should be manually checked every time a MAC spoofing or IP spoofing attack cannot be blocked through the method described in the next chapter.

Another point that has to be taken into account is the fact that Layer 2 devices include, for example, switches or APs. They have different characteristics and consequently they must be treated differently. In this project, the procedure to discover networks in cases that switches are the only Layer 2 devices present in the network was studied in detail. If there is any Layer 2 device of a different type, the steps to be followed should be the same as it was for switches. The only difference could be on the MIB object that must be retrieved because, as previously said, different network equipment can have different MIBs.

Finally, it is important to refer the case of routers using a switch module. Although they are routers by default, they can work like switches and have exactly the same behavior. They can also be accessed via SNMP and its information can be obtained similarly to any other network device. But during this project it was seen that most of these devices have a lack of information on their MIBs, which do not allow retrieving the necessary information from this type of devices as it is done for normal switches. For this reason, any router using a EtherSwitch card will be considered as an unmanaged switch.

4.3 Algorithm Implementation

Now that a complete description of the developed method and its limitations was made, it's time to explain how it was actual implemented in practice. Since the basis of this dissertation is the development of a few scripts to perform specific network monitoring tasks, the choice of the programming language was a crucial part of the project planning. The used operating system was Ubuntu and high-level and scripting languages are the most appropriate languages for the purpose of this work because interaction with the hardware and memory is not needed, they are object-oriented and provide an abstraction level that turns the scripting simpler and more robust [57]. The most commonly used scripting languages, and those that were

considered as options, were Bash, Python, Perl, Ruby and Javascript. For antiquity and compatibility reasons, Bash was the chosen scripting language. Bash is a Unix shell for the GNU project, which means that this language is at the same time a command interpreter that provides the user with an interface to interact with the operating system and a programming language with its own syntax that allows a user to write scripts with the ability to read commands directly from a file [58, 59]. As a consequence, Bash can be considered not only a scripting language like all the other mentioned languages but also a command line interpreter (shell) for GNU operating systems (UNIX-like computer operating systems) and the developed algorithms were based on this language.

As it has been said, the network discovery algorithm retrieves information from the different network devices. This information goes from the characteristics of the devices to network information like forwarding tables and ARP tables. In order to record and maintain the information for a posterior use, a database system was used. From the available options for database systems, only SQL databases were considered due to ease-of-use, support and administration reasons [60, 61]. In particular, the two most popular and used open source database systems are PostgreSQL and MySQL. From these, the choice for the deployment of this project was MySQL [62, 63]. MySQL is considered a fast, reliable and easy to use database system and it runs on a server providing multi-user access to a number of databases. Thus, it is commonly used on the development of web applications by making part of LAMP, the software bundle used for web development that also includes Linux, Apache and a high-level programming language like Perl, PHP or Python. MySQL is used by several high-traffic websites to perform data storage and logging of user data, which is a signal of its reliability. Referring to administration tools for MySQL, phpMyAdmin is one of the most used tools. phpMyAdmin is a free software written in PHP to manage MySQL database systems by providing an intuitive web interface with the ability to directly execute SQL statements and import or export data in different formats [64]. For the purpose of this project it was decided to use a MySQL database system instead of PostgreSQL due to its really well-supported documentation and reference manuals and due to the reason that it is the widely used database system for web development, allowing the project to be easily improved in the future with new features. phpMyAdmin was also adopted to manage and administrate the data retrieved from the network devices.

To have a complete network discovery algorithm, many scripts were developed. To start, we created a Bash script where the user provides all the information needed during the discovery process. Thus, this script doesn't execute any action over the network and was merely developed to introduce necessary information. First the user introduces the IP address of any interface of any router present on the network. This IP address is used as one of the parameters in the SNMP commands executed from the local machine and it works as a starting point to access one of the network devices. Later, the algorithm will use the information of this router to move to other devices. Then, it is asked if the user wants to discover the whole network or not. The router IP address and, if case, the stopping network are all the necessary information about the network.

Then, MySQL account information is required to allow the algorithm to create and manipulate a database where the information will be recorded. As will be seen, we created a script to automatically perform this task, so the next step is to introduce the username and password from a MySQL account that the user had already created. Finally, SNMP information is needed. SNMP provides the remote interaction between the local machine and the network equipment. The local computer will work as the manager or client, while each

network device works as an agent which means that they must be configured as an SNMP server. The algorithm was developed in order to work with both versions 2 and 3 of SNMP. This gives the user the freedom to choose the version that better fits his interests. So, the user should insert the SNMP version that it was configured on the network equipment to provide compatibility with the SNMP commands executed from the local computer. If the user chooses the version 2 of SNMP, then the community string configured on the devices should be introduced. Otherwise, in case of version 3, the username and authentication password are required. After all the previous information have been requested and introduced by the user according to his network and his MySQL account, another script will be executed in order to perform the complete network discovery.

Thus, two Bash scripts were created to perform the actual discovery of the network. These scripts have exactly the same content, except on the format of the SNMP commands. One is directed to SNMP version 2 and the other to version 3. The SNMP commands format is as follows:

- Version 2: snmpwalk v2c c [Community String] [Host IP Address] [MIB Object];
- Version 3: snmpwalk -v3 -u [Username] A [Password] -l authnopriv [Host IP Address] [MIB Object].

The correct script is selected according to the SNMP version introduced by the user. As it can be seen on the above commands, in the first script it was already provided the necessary information for its execution, with the exception of the MIB object, which is mainly what this script will be dealing with. The methodology described on the first section of this chapter will be executed in practice by this script. Many auxiliary text files are used during the execution of this script, thus, the first step is to read the information introduced on the previous script and delete old information that could be contained on these text files. Then, a function called *AnalyzeDevice* is executed. This function will be executed each time a new device is analyzed. It starts by verifing the type of the device (router or switch). There are many ways to identify the device but the chosen one was through the MIB object *sysObjectID* (OID .1.3.6.1.2.1.1.2), which returns a specific number sequence that identifies the device [65]. As it can be seen, in practice, the "snmpwalk" is the SNMP command used to obtain this information and it will also be used to retrieve most of the information contained in each device MIB. However, it returns more information than what is necessary. For this reason, the "snmpwalk" command is combined with a "cut" command to exclusively select the useful information. According to the type of device, the corresponding function is executed. In case of a router, the first step is to retrieve all the necessary MIB information from the device by executing a function called *Device_Router*. These MIB objects were described in the first section of this chapter and they are represented in Table 2.2 and Table 2.3. Since each one of these objects returns a list of results, this information is written into arrays.

The next steps consist of writing this information into temporary text files so it can be recorded later on the database and printed in a readable way for the user. This way, when running the algorithm, it is possible to immediately consult the routing table, ARP table and IP addresses from all the interfaces of the router. Then, with all information saved on auxiliary files, it is necessary to move to another network device. The method was already explained in detail, so it's easy to follow the proceeding by reading destination networks, routing types and so on, from the arrays. In case of an indirect destination network, when the next-hop IP address has not already been analyzed, the IP address of the current router is

recorded, a variable assumes the value of that address and function *AnalyzeDevice* is executed. This way, SNMP commands will be sent to this new host. When the function returns to the first router, its recorded IP address is restored and function *Device_Router* is executed so the MIB information is available again.

For the case of directly connected networks the steps were also described in the first section and it is only necessary to use information contained in the arrays. When moving into a Layer 2 device the procedure is the same of the routers with only one difference. The MIB object *ipNetToMediaNetAddress* contain IP addresses from routers, Layer 2 devices (managed and unmanaged) and end hosts. When executing a SNMP command to an unmanaged device or end device it will return an error and it will try to access it again periodically. To avoid this situation and because we are not interested in these devices, the solution is to send a "snmpget" command with a random MIB object (*sysDescr*, in the case) for the IP address of the unmanaged device and wait 1 second. If after this second there is an answer, it means that we are ahead of a managed Layer 2 device. Otherwise, it is an unmanaged device or an end device and this IP address is simply skipped. This could be a rough way of dealing with this situation but it is actually efficient and that's why it was kept like this. When moving to a Layer 2 device the current IP is saved and then restored when we come back. Also the *Device_Router* function is executed again. For the case the network device is identified as a switch, function *Switch* is executed. As in case of routers, the first step is to retrieve the necessary information from its MIB, which is done with function *Device_Switch*. The MIB objects were represented on Table 4.1 and the proceeding to select useful information was also described. Information about the device and its MAC Address Table is written into text files to be recorded later and the table is organized and printed to be readable when the algorithm is executed. Then, the algorithm returns to the previous router and the process continues.

When function *AnalyzeDevice* returns to the first analyzed router and reaches its end, two other scripts are executed. These are two PHP scripts developed in order to save information in a database. The first one is dedicated to the creation of the database and subsequent tables. It uses the MySQL account information provided on the first script to connect to the database server. The database server address was maintained as default (127.0.0.1) and the name of the database was simply defined as "network". It was decided to define the same name to any user to keep the algorithm as simple as possible and also because there are no advantages of changing it. Using a SQL command the connection is established. After connecting to the database server, the script will check if a database with the name "network" already exists. If not, a new database is created. Otherwise, it will delete it and create a new one. This is a rough but efficient way to ensure that only updated information is available on the database, while the old one is released. After the database has been created, tables where information will be recorded are created . A table with information about the devices is created, a Routing table, an ARP table, a MAC Address table and a table with all the IP addresses from each interface of the routers. Thus, this first PHP script has simply a supporting functionality for the creation of databases and tables.

The other PHP script was developed to fill the tables with the information retrieved from the network equipment. Again, the connection to the database server is established with the provided username and password and the "network" database is selected. As it was said, all information retrieved from the devices was written in text files. So, the first step is to read all auxiliary text files with information about the devices and insert it in the devices table. This table will allow to identify and to distinguish all network equipment present on the network. Then, for each one of these devices the text files containing the information associated to

them will be read . Finally, after all data has been inserted on the correct table, all auxiliary text files containing information are deleted. In the first section of this chapter we made a general description of this method for network discovey and now we described the practical implementation of the whole algorithm.

In summary, the first 3 scripts combined form an efficient algorithm that uses the SNMP protocol to discover all network equipment and retrieve useful information. Then, the last 2 scripts record this information into a database so it can be available for monitoring tasks. During this chapter, the developed method for network discovery was presented in detail, as well as the limitations of the algorithm and how it was implemented. After running this algorithm, any network equipment that supports SNMP is known and we can now move to the next chapter where we will describe the method to detect spoofing attacks.

Chapter 5

Developed Spoofing Attack Detection Algorithms

The previous chapter described a method to discover all network devices present in a network, in particular, Layer 2 and Layer 3 devices that support SNMP. After running the network discovery algorithm, all network devices become known to the local machine that is monitoring the network. This allows to perform other monitoring tasks.

The objective of this project is the development of several methodologies for network monitoring and the following approach that will be presented is focused on network security and protection. Most networks are exposed to a variety of malicious activities, so protecting them against these security threats is essential for a secure and trustful system. A definition of spoofing, which is a common type of network attack, was already given in a previous chapter. Some approaches to protect the system against these threats were also discussed. This chapter presents a methodology, which uses a distributed approach, for the detection of spoofing attacks, in particular MAC and IP spoofing attacks. As in the network discovery algorithm, SNMP is also the basis of the proposed method but, due to different characteristics of both attacks, different procedures should be taken in order to detect them. A solution to block these attacks when they are detected will also be discussed.

Before presenting the method, it is important to make some considerations about it. The first one has to do with Layer 2 devices and its characteristics. In case of switches, they have different ports and each one is used by a unique device to connect to it. On another hand, an AP has a wireless interface that is used by many devices at same time. Thus, for the proposed method we will describe in detail the steps for the detection and blocking of spoofing attacks, in case the attacker is accessing the network from a switch. In this case, the port where he is connected to must be blocked. If we are dealing with an attack triggered from an access point, then the attack can only be detected when it belongs to the IP spoofing attack type. This is due to the fact that, using this method, MAC spoofing attacks are detected based on the MAC address and interface that the intruder is using to access the network. In case the attacker is accessing the network from the same access point of the authorized client, there is no way to distinguish between them because they are using the same MAC address and the same interface. Thus, when performing MAC spoofing detection, APs are considered unmanaged devices. That situation does not happen on IP spoofing attacks because in this case the task is to find similar IP addresses and once this happens, the MAC address of the intruder is immediately found. Then, the problem consists only of finding it on the network

and blocking it. If the attacker is accessing the network from an AP, the procedure is similar to the switches' case but, instead of blocking the interface that the attacker is using (the wireless interface), the MAC address of the host he is using to perform the attack is blocked; otherwise, the other devices that are using the interface could not access the network anymore. Blocking the access of a MAC address of a certain end host to an AP must be done manually via SSH, for example, through the MAC ACL of the AP.

The other limitation of this method is the fact that, when discovering the network, there could be some unmanaged devices, as was mentioned in the previous chapter. This means that if the attacker is accessing the network from one of these unmanaged devices, the following method will be able to detect the attack but it won't be able to block it. As a consequence, if a spoofing attack is detected but it's not blocked, the user must check manually all the unmanaged devices because most certainly the intruder is performing the attack from one of these hosts.

This chapter is divided into two sections; one directed for MAC spoofing attacks and the other to IP spoofing attacks. In both sections, the methodologies and algorithms developed for detection and consequent blocking of the corresponding spoofing attacks will be presented. Then, the algorithms implemented in practice will also be discussed, with a detailed description of all steps.

5.1 MAC Spoofing Detection

As previously said, Layer 2 devices use MAC addresses as their LAN identifiers. This address is assigned by the manufacturer to each interface of the device and is controlled by OUI to be globally unique for all LAN-based devices. But MAC addresses can easily be changed in most devices without any consequences on their performance. This means that faking MAC addresses is a simple way for an attacker to perform network security attacks. There are several reasons to perform this kind of attacks [66], but one of the most common is to impersonate an already authenticated user. In this case, the attacker just needs to know the client MAC address and change its own address accordingly. In this way, and since the user is already authenticated on the network, the attacker can send and receive traffic disguised by the MAC address of the user. Next, we will present a procedure, based on the SNMP protocol, to detect these Layer 2 attacks and block the access of the intruder to the network.

5.1.1 Attack Detection

In Figure 5.1 the method to detect and block MAC spoofing attacks is described. This mechanism will basically create a record of the MAC addresses of all interfaces of the different network end devices. If someone tries to fake a MAC address, then the port or even the switch will change when, compared with this record, because that MAC address will appear on another location. This algorithm is able to detect such situation and figure out if it is really a MAC spoofing attack or if the client has simply changed the physical location of the device.

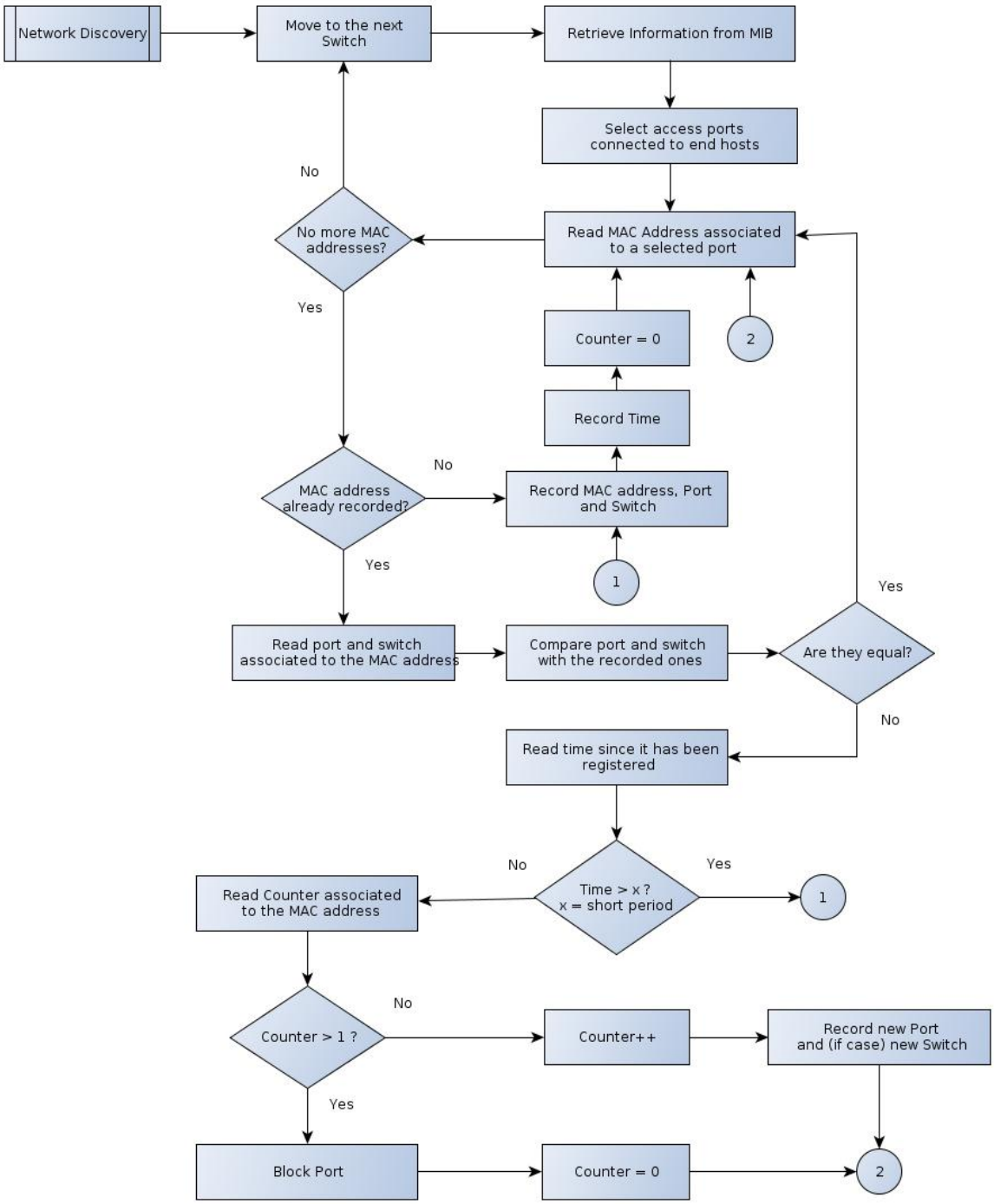


Figure 5.1: MAC Spoofing Detection and Blocking

The algorithm starts by performing the network discovery procedure described in the previous chapter in order to find all network devices and identify them. When dealing with MAC spoofing attacks, we just have to deal with MAC addresses and, therefore, only Layer 2 devices (switches, in the case) need to be analyzed. After selecting these devices, each one is analyzed individually. Then, useful information is retrieved from the MIB of the switches. The data

for the detection of MAC spoofing attacks should be selected and retrieved using the SNMP "snmpwalk" command and then put in an array, so it can be easily accessed. The necessary MIB objects are represented in Table 4.1. The *dot1dTpFdbAddress*, *dot1dTpFdbPort* and *atPhysAddress* objects were all already mentioned in the previous chapter and they provide information corresponding to the MAC Address Table of the switch. Below, we will show why this information is so important and we will mention other MIB objects that are used in this detection method.

On the switch, only access ports are important because end hosts are connected there. Since all ports are already known, access ports can be selected using the MIB object *vlanPortIslOperStatus* (also in Table 4.1), which returns the value '1' for Trunking or '2' for Not Trunking. However, an access port can also be connected to another network device instead of an end host. In this case, the MIB object *atPhysAddress* should be used. If any of the MAC addresses associated to an access port belongs to the list of MAC addresses of the *atPhysAddress* object, it means that the access port is not connected to an end device and it should be excluded from the ports to analyze. Figure 5.2 represents an example of the access ports that need to be analyzed (those connected to end devices) and the excluded ports.

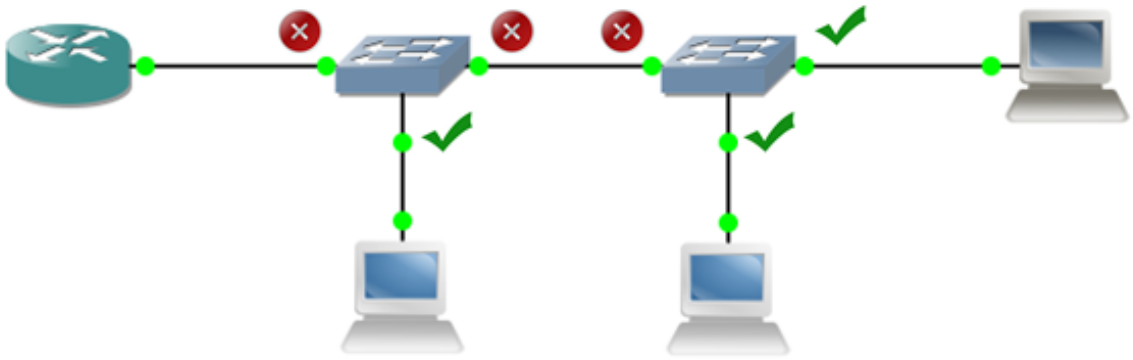


Figure 5.2: Access ports selection

The first stage of the procedure is completed and we have now all the necessary information from the switch, with a list of all MAC addresses of end hosts connected to the switch as well as the access ports where they are connected to. The next step consists on reading each MAC address associated to the selected access ports. When a MAC address is analyzed, the algorithm should check if it was already recorded. It was chosen to maintain a record of all MAC addresses of the end hosts that are found on the network. If the MAC address that it is being analyzed does not exist yet in this historic, then a record must be added, containing the MAC address, the corresponding network device and the port where it is connected to. The access port is already known and the information about the switch can be retrieved through the MIB object *hostName* (OID .1.3.6.1.4.1.9.2.1.3). The registration time is also recorded, as well as a counter whose value is '0'. Figure 5.3 represents this procedure. This is all the information that is needed regarding each MAC address that is detected in the network. Then, the next MAC address in the array of end hosts' MAC addresses should be read. When there are no more MAC addresses to read, the algorithm moves to the next Layer 2 device.

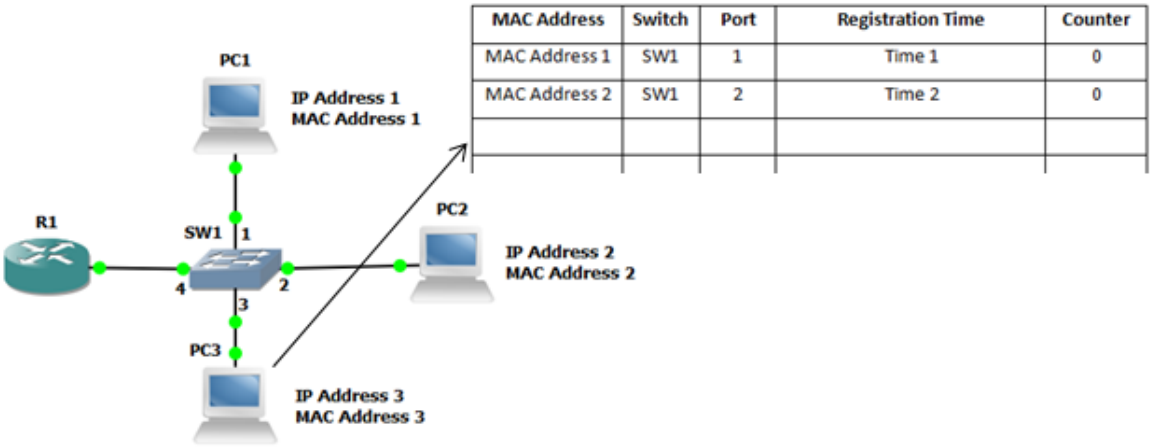


Figure 5.3: End host registration process

When a MAC address is already registered, its location in the network should be checked to verify if it is in the same place or if it has moved to another location. The historic already contains information about the switch and port associated to this MAC address, which allows identifying the location of the end host. So, the recorded information is compared to the switch and port that the MAC address is using now: if they are equal, it means that the end host is in the same place (Fig. 5.4); otherwise, we can be sure that the end host has changed its physical location (Fig. 5.5) or someone is faking this MAC address and is using it to connect to the network from another location (Fig. 5.6). When the second case is detected, we are ahead of a possible MAC spoofing attack. In the next section we will describe the procedure to evaluate and determine if someone is trying to perform a network attack and, if it is the case, to block the access of this intruder.

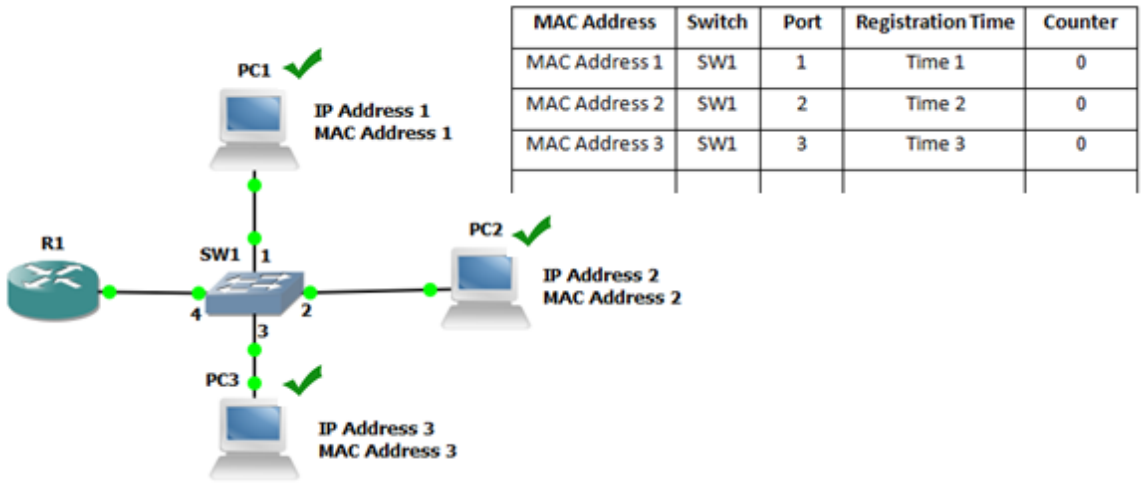


Figure 5.4: No changes on the network

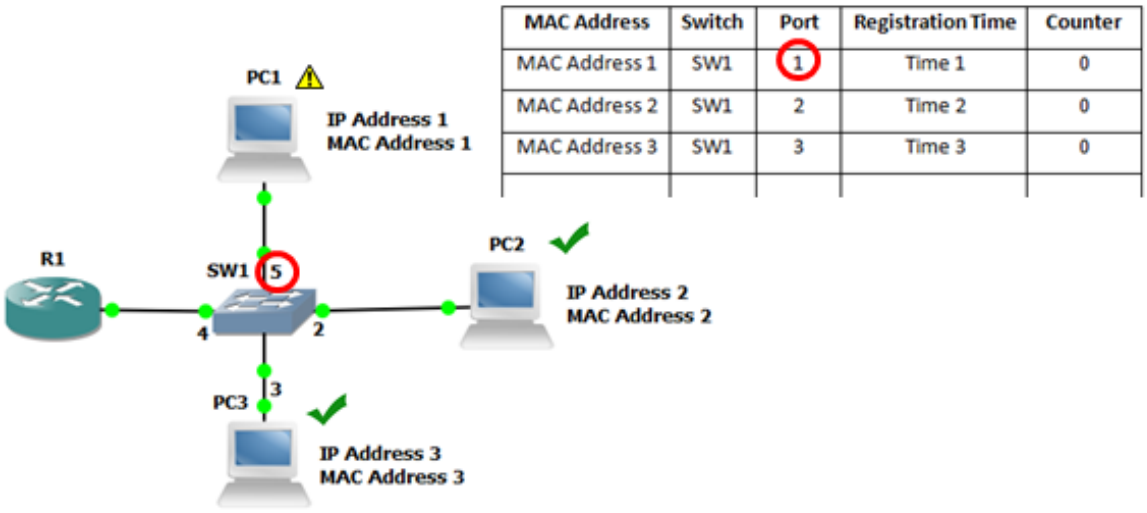


Figure 5.5: End host changed physical location

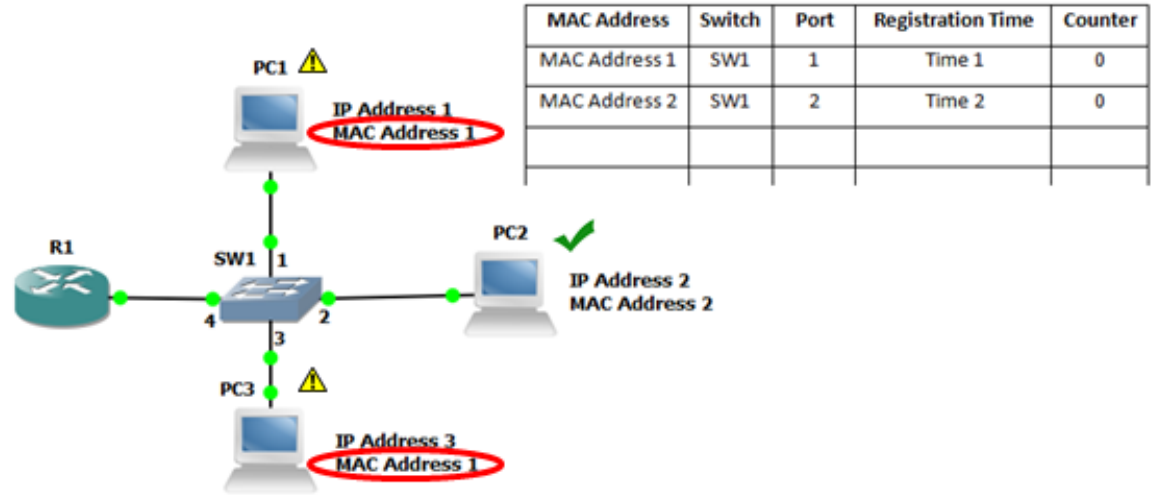


Figure 5.6: MAC spoofing attack scenario

5.1.2 Attack Blocking

Once a possible attack is detected, it is important to verify if it is a real attack or if the user has just moved the end device to another location on the network. The first question that should be answered in order to understand the reason for this change is: how much time has passed since the MAC address has been registered? When the MAC address was recorded for the first time, many parameters were saved and one of them was exactly the registration time. In this way, it is possible to check how much time has elapsed since that instant. When there is a MAC spoofing attack, a client is communicating and the attacker is using the same MAC address to send and receive traffic from the network, but from another location. This means that in a real MAC spoofing attack changes will be detected in the port (and possibly

in the switch) associated to the MAC address in a short period (of a few seconds). So, if the time elapsed since the MAC address has been registered is greater than this short time interval, it means that probably the client has just changed his host location and the network is not under attack. In this case, the new port has to be recorded and, if it is the case, the new switch. The registration time is also updated and the counter is set to value '0' (if it was not '0' already). Figure 5.7 shows this situation.

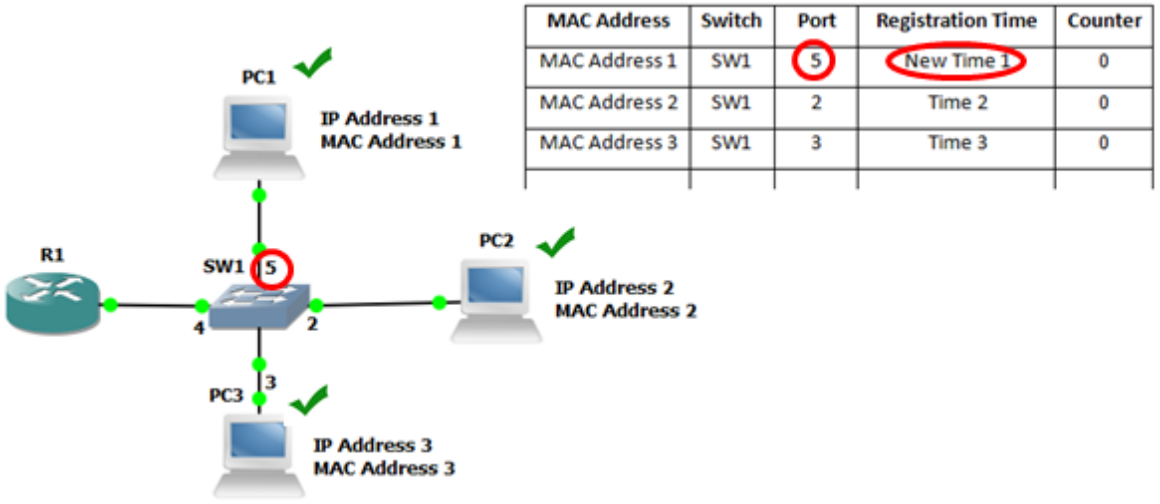


Figure 5.7: End host information updated

On the other hand, if the time since the MAC address registration is shorter than the time period that is considered normal when the network is under attack, then another question arises: how many times this MAC address has changed its location during the short time period we are considering? The counter parameter can be used to answer this question. If a change was detected in the last seconds, then the counter associated to the MAC address must be checked. If the counter has the value '0' or '1', then it means that in the last seconds that MAC address has not changed its location or has changed it only once, which can be considered normal. In this case, the counter is incremented and the new port is updated. The time parameter is not updated because it is necessary to check if there will be more changes in the next few seconds (Fig. 5.8). If the counter reaches a value greater than '1', it means that a change of location was detected more than once in a short period of a few seconds, which can be considered as an unusual behavior and consequently there is a high probability that the network is under a MAC spoofing attack (Fig. 5.9).

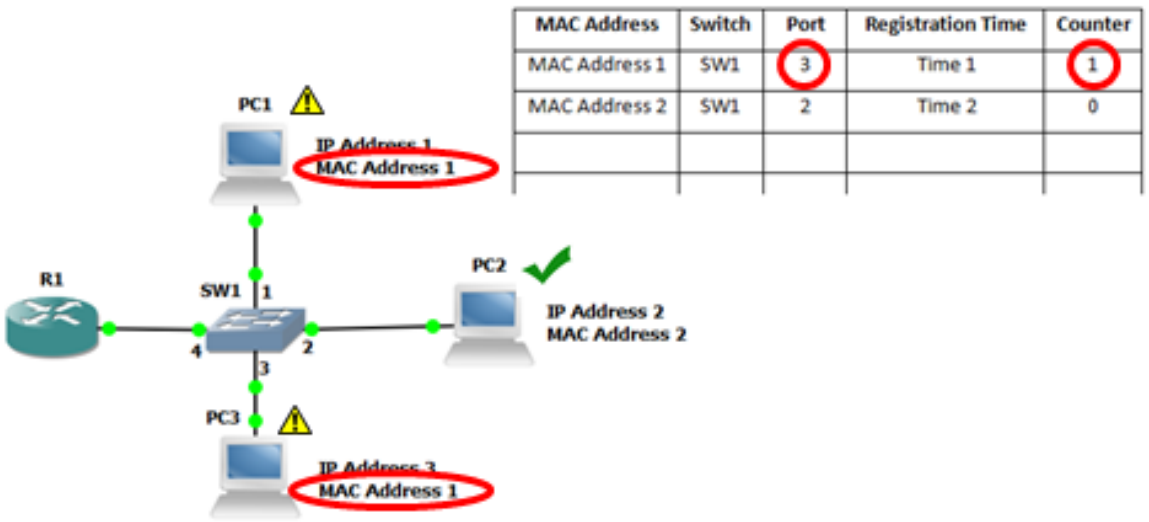


Figure 5.8: MAC address detected on different location - Counter incremented

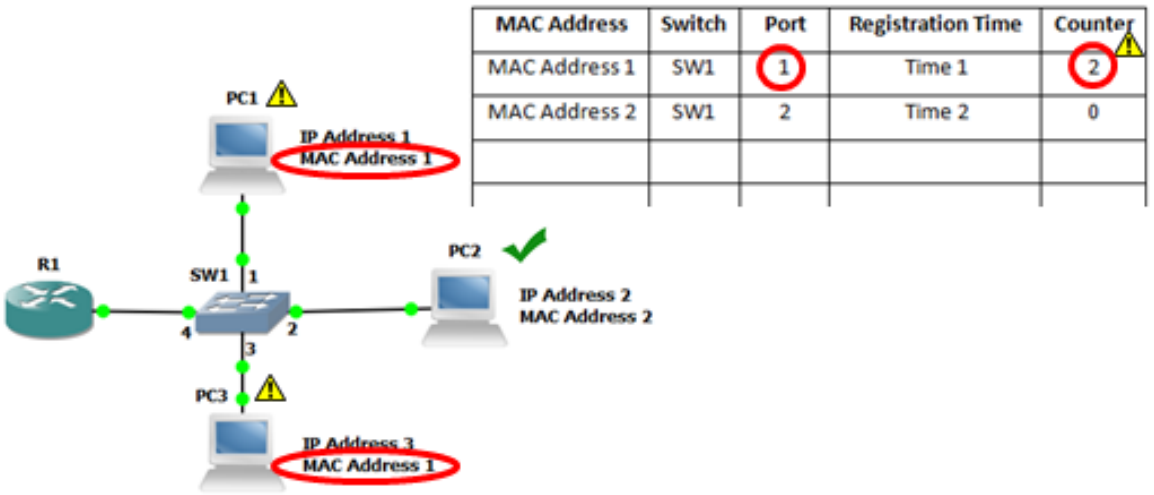


Figure 5.9: Counter reaches value '2' - MAC spoofing attack detected

When a MAC spoofing attack is detected, it must be blocked. Using this method, this operation is really easy to accomplish because a record of the previous ports and switches is maintained and compared to the port and switch that a given MAC address is using now to access the network. So, if a MAC spoofing attack is detected and the attacker is using a switch to perform the attack, the port where the MAC address is connected right before the counter reaches the value '2' will be blocked. Using the information corresponding to the bridge ports associated to the different MAC address (available from the *dot1dTpFdbPort* MIB object), the interface index of the device that has to be blocked can be retrieved using the SNMP command "snmpget" over the *dot1dBasePortIfIndex* MIB object (OID .1.3.6.1.2.1.17.1.4.1.2). Finally, it is possible to block the port using the SNMP command "snmpset" over the MIB

object *ifAdminStatus* (OID .1.3.6.1.2.1.2.2.1.7), which will shut down the interface and block the attack. The legitimate user continues accessing the network without any problems (Fig. 5.10). In case the attacker is accessing the network from an unmanaged device, the device must be checked manually, as previously said.

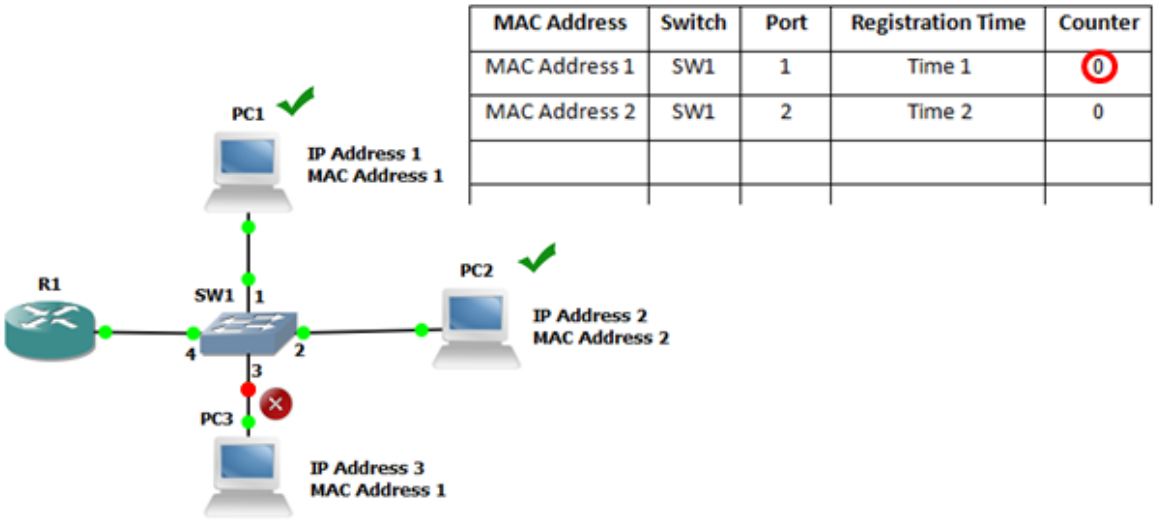


Figure 5.10: MAC spoofing attack blocked

5.1.3 Algorithm Implementation

In the previous sections of this chapter we have presented the general algorithm for detection and posterior blocking of MAC spoofing attacks. This method can be developed in practice on different ways according to the system where it will be deployed. Thus, this section describes one practical implementation of the previous algorithm for the purpose of this project and for the creation of testing scenarios that will be used later.

For the implementation of this algorithm, a single script using Bash language was developed, like we have done for the network discovery algorithm. The reasons for the use of this scripting language are the same that were presented before. In terms of the contents of this script, it starts by requesting the the SNMP version from the user. Obviously, this version must be the same as the one configured on the network equipment. Depending on the SNMP version, the user should insert the community string or the username and password. Also possible temporary files containing old information from previous executions of the script are deleted. Then, information with all Layer 2 devices, in particular switches, obtained from the execution of the network discovery algorithm is read and the script is ready for detection of MAC spoofing attacks by executing function *Detect_Spoofing* in an infinite cycle.

For the detection of the attacks, the function accesses sequentially each switch by using their managing IP addresses. These IP addresses are used mainly to retrieve information from the switch MIB. For each switch, a function *Switch* is executed and "snmpwalk" commands are sent to retrieve information from the *dot1dTpFdbAddress*, *dot1dTpFdbPort* and *atPhysAddress* MIB objects (Table 4.1) and put it in arrays. These objects provide informa-

tion about the MAC addresses of the forwarding table of the switch and the corresponding bridge ports, as well as the MAC addresses from all network equipment on the network. For the detection of MAC spoofing attacks through this method, a record of all end hosts connected to each switch will be created. So, since the MAC addresses of all devices in the network are already known, it's necessary to select only end devices. For each bridge port, it is used the *vlanPortIslOperStatus* MIB object to select the access ports. Then, for each access port, it is verified if the MAC addresses associated to that port belong to the list of MAC addresses retrieved from the *atPhysAddress* MIB object. It is known that end hosts are not present on this list (only network devices are) and so, only MAC addresses that are not contained on this list are selected. After executing this task, all end devices connected to the switch and the bridge port where they are connected to are known.

The next task is to execute a cycle to analyze each of the selected MAC addresses. For each one, it is checked if it was already registered. To keep a record of all MAC addresses and other useful information about an end device, some auxiliary text files are used. So, the auxiliary file containing the MAC addresses is read to verify the presence or not of each MAC address. In case of a new one, the MAC address is added to the file and information about the port and switch where it is connected to, registration time and counter must be recorded. The port is read from the array with the selected ports and written into the corresponding text file. For the identification of the switch, its hostname is used, which is retrieved from the *hostName* MIB object and written into the devices text file. For the registration time, the command "date" is executed with the following parameters to consult the present time: *-H* for hours, *-M* for minutes and *S* for seconds. Then, hours and minutes are converted into seconds and the registration time is written into the correspondent text file as the total number of seconds. It was decided to record also the present date, so the script can be more accurate in cases that a change is detected after a few seconds since the device registration but on different days. It's a rare situation but it has to be considered. The date is obtained from the "date" command too, with parameters *m*, *d* and *Y* for month, day and year, respectively. This information is written in a specific text file. Finally, for the counter it is simply written the value '0' into a counters' text file.

On other hand, if a MAC address is already present in the MAC addresses text file, it must be checked where it was connected when it was recorded. This is done by reading the corresponding port from the text file containing this information. An auxiliary variable was used to know from which line the MAC address was read, so the port information should be in the same line. When the line with the port associated to the MAC address is read, it is compared to the present port contained on the array of selected ports. If they're different, there was definitely a change on the origin of the MAC address and function *Possible_Spoofing* is executed. Otherwise, the same line is read from the file containing the hostname of the switches to check the device where that end host is connected to. Even unlikely, there could be the case in which the port associated to an end device is the same as the recorded one but from different switches. If the hostnames doesn't match, the *Possible_Spoofing* function is also executed.

As it was explained on the previous sections, when a change on the origin of a MAC address is detected, it must be checked the time elapsed since the registration of that end host. Starting from a more general perspective, the line from the dates text file corresponding to the registration "date" of that MAC address is read. Using the date command with the previous parameters, the present date is compared with the recorded date. If they're different, it is automatically assumed that the change happened on different days and there was simply

a change on the device location, even though the attack could be performed on a day change (23:59 to 00:00), which is very unlikely. So, in this case, the new device information is registered. The exact line from the files that must be replaced with new information is known, and the "sed" command allows performing this task. To facilitate the task all information is updated, even if certain parameters haven't changed. So, the port and switch where the device is connected now is written in the corresponding text files by replacing the previous information. Also the new date and time is read and written into their files as well as the counter with the value '0'.

In the situation that dates are equal, the line with information about the registration time of that device is read. Then, the present time is read by executing the "date" command and converted into seconds. The time of registration is subtracted to the present time and the difference between both corresponds to the time elapsed. As previously mentioned, in this method, MAC spoofing attacks are detected based on consecutive changes on the origin of a certain MAC address during a short time period. Thus, for the implementation of the algorithm, this time period was defined as 30 seconds. So, if the difference between registration time and present time is greater than 30 seconds or a MAC spoofing attack has just been blocked, new device information should be recorded in substitution of the old one. In the first case it is assumed that the end host changed to a different location on the network and the second case is due to the fact that when an attack is blocked, information about the intruder's device is contained on the auxiliary text files and this information needs to be replaced for the authentic and legitimate client information. This is done exactly as previously by executing "sed" commands. In the second case, it is also necessary to set the variable signalizing that the attack has been blocked with value '0'. On other hand, if the time passed is shorter than the defined 30 seconds and any attack wasn't blocked recently, the counter parameter associated to that MAC address must be checked. Again, the counters text file is read line by line until the line corresponding to the device information is reached. If the counter value is '0' or '1', the counter is incremented by executing the "sed" command to substitute it for the new value in the file and information about the switch and port where the host is connected is updated. Finally, the MAC spoofing attack is detected if counter value that was read from the value is greater than '1'. In this case, a *Block_Spoofing* function is executed.

To block the spoofing attack, the interface where the MAC address is currently connected will be simply turned down. The bridge port associated to the MAC address is known and the "snmpget" command is executed to retrieve the information about the corresponding port index. To turn the interface down, the "snmpset" command is executed. The counter value is set with value '0' using the "sed" command and the variable signalizing that an attack has just been block is set with value '1'.

We described the whole script developed to implement the algorithm described in the previous sections of this chapter. In a later chapter, some tests will be performed using this script. Next section will present the developed methodology for the detection of other type of network attacks: IP spoofing attacks.

5.2 IP Spoofing Detection

After the analysis of Layer 2 network attacks, it is time to take a look at Layer 3 attacks or IP spoofing attacks. Unlike MAC addresses, IP addresses must be configured whenever new equipment is connected to the network; otherwise, communication will fail. But, when

IP addresses are not assigned automatically through DHCP and the user doesn't know all IP addresses of the network, there is always the risk to configure a device with an IP address that is already in use. IP spoofing attacks are based on the principle that if the intruder impersonates an authorized client by using its IP address, then he can get access to the network because all devices will believe that those packets come from a trusted host [67].

There are several tools to prevent this kind of network attacks. Here, it will be presented a simple methodology based on the SNMP protocol. Like it was done in the previous chapter, the algorithm description will be divided into two parts: detect the IP spoofing attack and block it. Then, a section dedicated to the implementation of this algorithm will be presented.

5.2.1 Attack Detection

Figure 5.11 shows a method to detect IP spoofing attacks. For each detected end host a record is created containing its IP and MAC addresses. If an attacker tries to use an IP address that is already in use, that occurrence will be detected by the simple reason that the MAC address of his device is different from the MAC address of the victim. This is the basic principle of this method.

As shown in Fig. 5.11, the first thing to do is a network discovery to find all routers, switches and APs on the network. Since this method is about Layer 3 attacks, all routers must be analyzed until an IP spoofing attack is detected. When that happens, the attacker access to the network must be blocked. To do so, all Layer 2 devices have to be checked until the intruder is found.

First of all, after having a complete list of all Layer 2 and Layer 3 devices, each router of the network is analyzed separately. Then, it is necessary to retrieve and select information from its MIB in order to perform the attack detection. The MIB objects that we need to retrieve from routers and put in an array are: *ipNetToMediaNetAddress*, *ipNetToMediaType* and *atPhysAddress*. The correspondent OIDs can be consulted in Table 2.3.

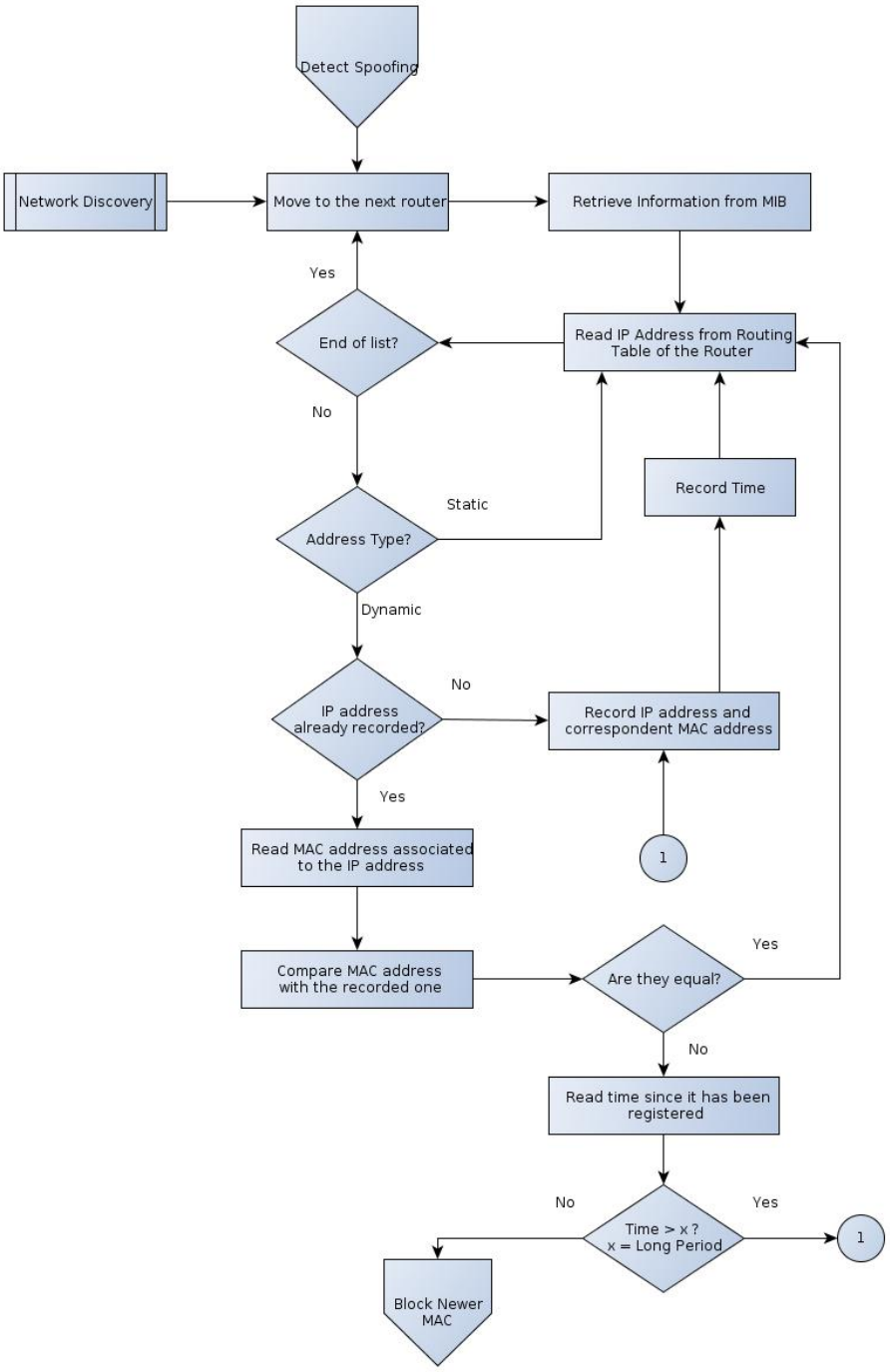


Figure 5.11: IP Spoofing Detection

With this information, it is possible to have access to all IP addresses of the router's forwarding table, as well as the corresponding MAC addresses and address types. A new cycle must be performed in order to analyze all these IP addresses, until there are no more addresses to read, and then move to another router and perform the same steps. When an IP address is analyzed, the first thing to do is to check for the address type. An IP address can be selected to be static or dynamic, but in this case we are only interested on dynamic addresses because we are looking for IP addresses of end devices and these are always dynamic. If an IP address is static, then the next IP address from the array must be read. If that IP address is dynamic, we have to check if it was already recorded. Like we did for MAC spoofing attacks, a record including some parameters is kept in order to have a comparison base for the future. For each end host IP address, the corresponding MAC address and registration time are saved. Figure 5.12 represents the registration process.

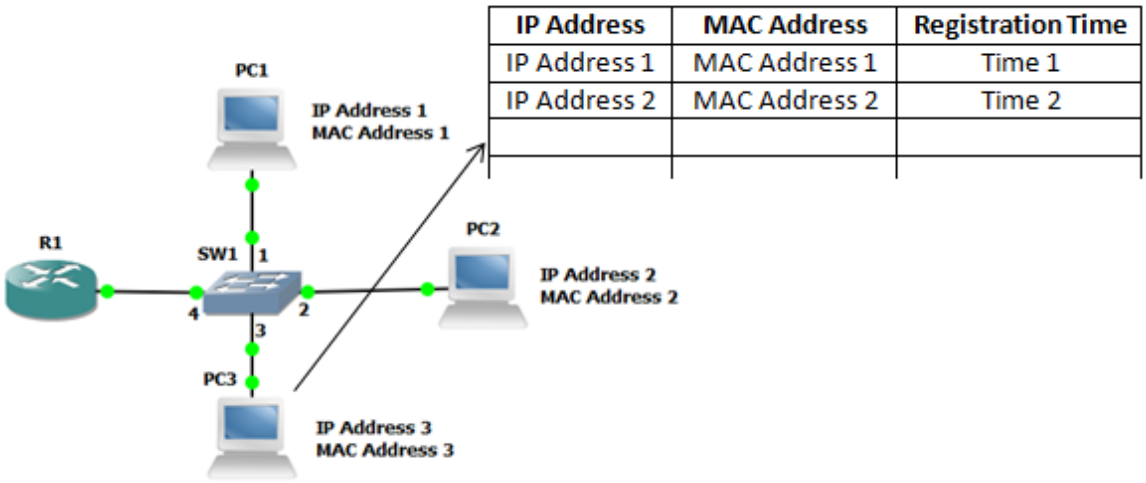
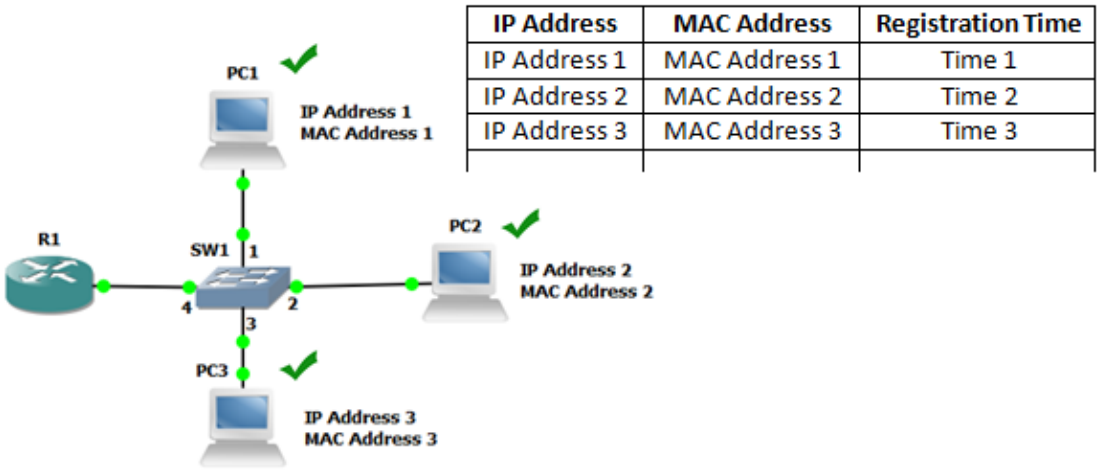


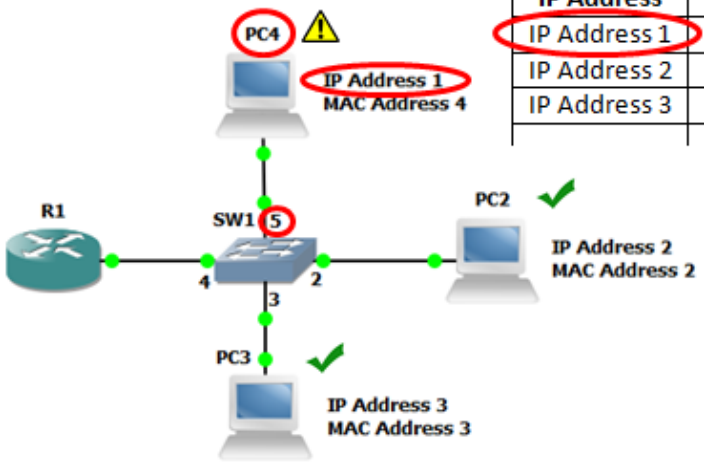
Figure 5.12: End host registration process

If a given IP address was already registered, then the recorded information must be checked. First, the IP address that was recorded should be read and compared to the MAC address of the device that it is using the same IP address at this moment. If they are equal, then it means that the IP address is being used by the same equipment and nothing wrong is happening, so the next IP address from the array can be read (Fig. 5.13). If the MAC address is different, then two things could have happened: the user simply started using a new device and configured it with the same IP address in order to have access to the network (Fig. 5.14) or someone is trying to perform a network attack using the IP address of an authorized client (Fig. 5.15).



IP Address	MAC Address	Registration Time
IP Address 1	MAC Address 1	Time 1
IP Address 2	MAC Address 2	Time 2
IP Address 3	MAC Address 3	Time 3

Figure 5.13: Same end hosts associated to IP addresses



IP Address	MAC Address	Registration Time
IP Address 1	MAC Address 1	Time 1
IP Address 2	MAC Address 2	Time 2
IP Address 3	MAC Address 3	Time 3

Figure 5.14: End host configured with an already assigned IP address

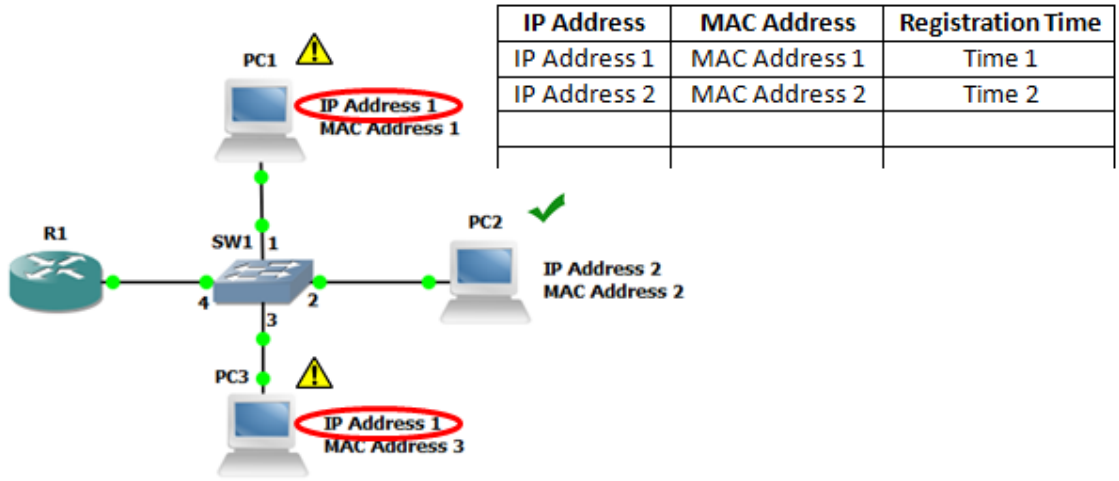


Figure 5.15: IP spoofing attack scenario

In order to distinguish between these two situations, the registration time parameter is used. It is not common that an IP address is associated to different end devices in a short period of time. It can happen occasionally, for example when an end host leaves the network and the IP address that was associated to it is available to be assigned to another device. It is expected that once an end host is configured with an IP address, no one else will get the same IP address for a period of time of at least some minutes. Based on this principle, if different MAC addresses are detected for the same IP address, it must be verified how many time has passed since it was registered. If this time is greater than the time period that is considered as normal, then the situation from Figure 5.14 is considered and a new record for this new MAC address must be created, besides updating the new registration time (Fig. 5.16).

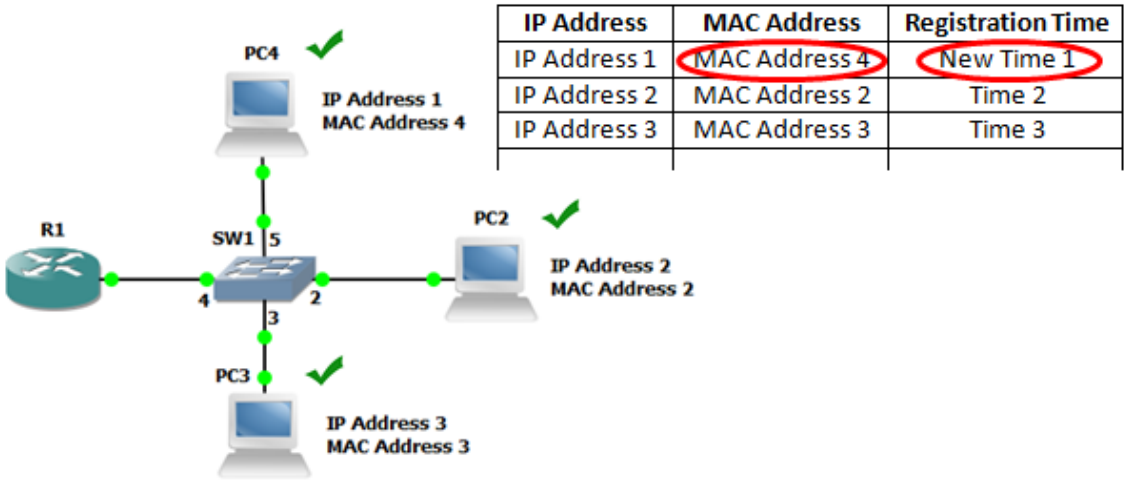


Figure 5.16: New end host information updated

On the other hand, if only a short period of time has elapsed since it was registered, then

there is a great probability that someone is using the IP address of someone else to perform an IP spoofing attack against the network. This is the case of Figure 5.15 and we have to move on to the next stage in order to find the location on the network of this new MAC address and block the port of the switch or AP where it is connected to.

5.2.2 Attack Blocking

At this point, the IP spoofing attack was detected and the MAC address of the device that it is being used to perform the attack is already known. So, each Layer 2 device on the network should be analyzed in order to localize this MAC address. Figure 5.17 describes the method that was devised to block IP spoofing attacks.

The first thing to do is to retrieve the necessary information from the MIB of each Layer 2 device, as was previously done every time we needed to analyze any network device. In this case, the MIB information that it will be used is the same that it was mentioned before to detect MAC spoofing attacks and it is present in Table 4.1. So, the MIB objects retrieved from switchs are: *dot1dTpFdbAddress*, *dot1dTpFdbPort* and *atPhysAddress*.

In the case of switches, ports that are being used exclusively by end devices should be identified (Fig. 5.2). In order to do that, the switch access ports are selected using the MIB object *vlanPortIslOperStatus* (OID .1.3.6.1.4.1.9.5.1.9.3.1.8). Then, the ports that are connected to other network devices must be excluded. If the MAC address associated to any of these ports is present in the list of MAC addresses retrieved from the *atPhysAddress* MIB object, it means that this port is not connected to an end host and it can be excluded. After performing these steps, we have only the necessary switch ports.

The next step is to analyze each one of the selected ports until there are no more ports to read and, then, move to the next Layer 2 device. For each port, the associated MAC address in this particular moment is read. Then, this MAC address is compared with the MAC address that was identified as belonging to the intruder. If they are different, it means that the end device that is connected to the port is not the one that we are looking for and we can move to the next port. When the right MAC address is finally found, the associated port is blocked (Fig. 5.18). The interface index is necessary to block the port. Using the bridge port retrieved from the *dot1dTpFdbPort* MIB object, it is possible to get the corresponding interface index using the *dot1dBasePortIfIndex* MIB object and executing a "snmpget" command of the SNMP protocol. To turn the interface down, the "snmpset" command is executed over the *ifAdminStatus* MIB object (OID .1.3.6.1.2.1.2.2.1.7).

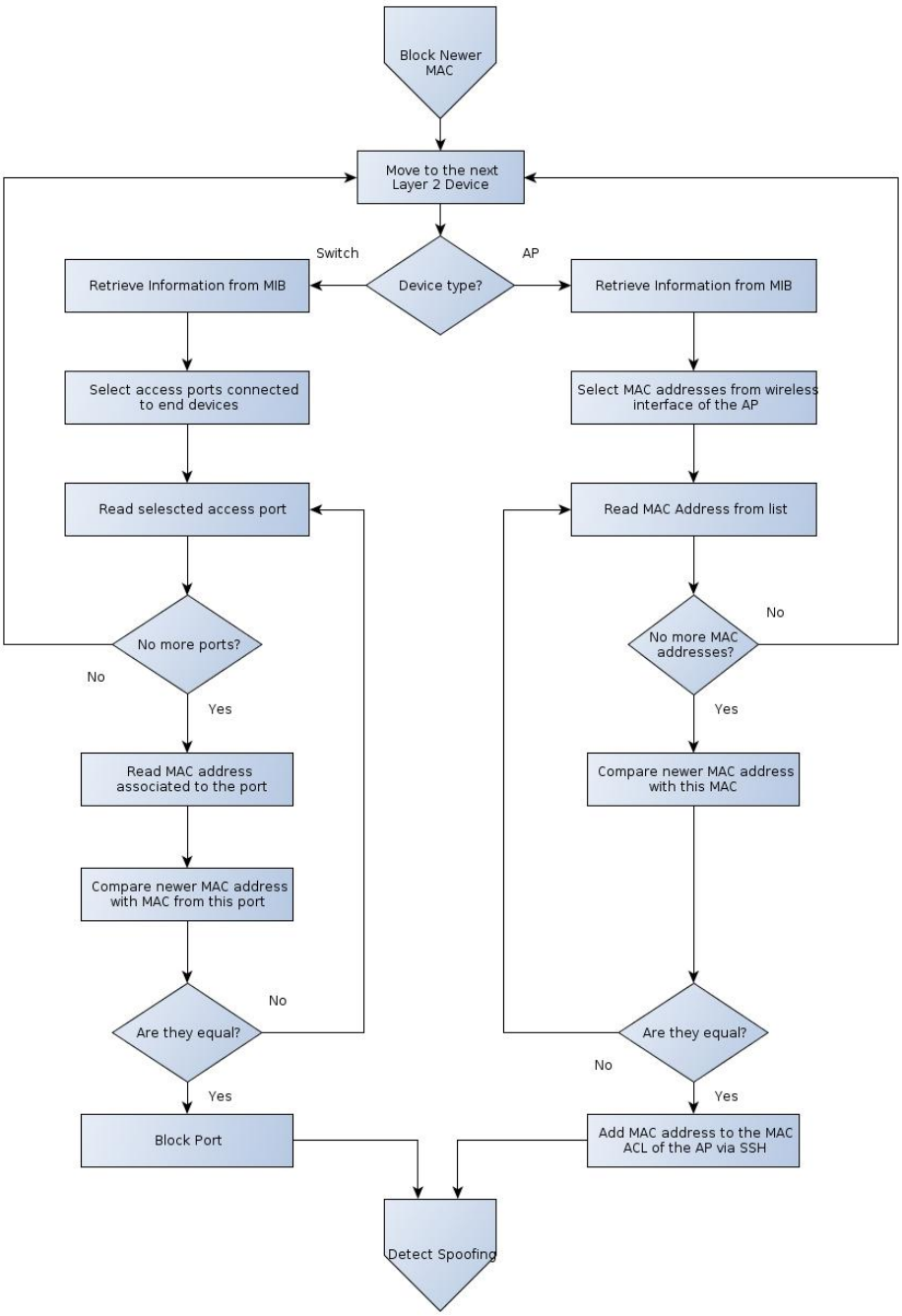


Figure 5.17: IP Spoofing Blocking

In case the attacker is accessing the network from an AP, all MAC addresses connected to the wireless interface will be read. If the MAC address of the intruder is not present on this list of MAC addresses, it means that it is not connected to the access point and we can move to the next Layer 2 device. Otherwise, if the MAC address we are looking for is detected in a certain AP, it must be added to the MAC ACL of the access point via SSH in order to block the access of the host to the network.

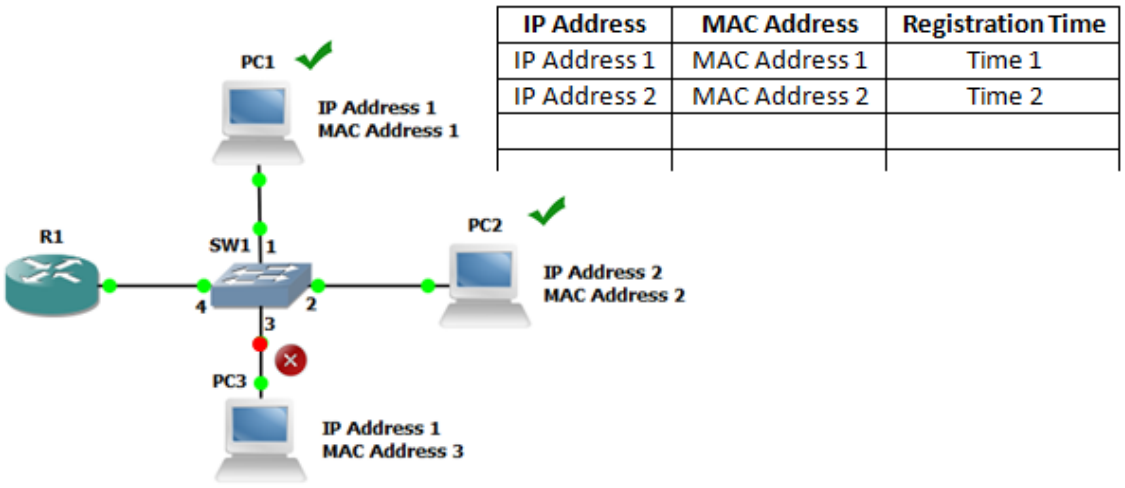


Figure 5.18: IP spoofing attack blocked

This methodology is an efficient way to block IP spoofing attacks from intruders that are accessing the network using switches or access points.

5.2.3 Algorithm Implementation

Finally, a practical implementation of the previous algorithm will be described. Like in the case of MAC spoofing attack detection, a single script using Bash scripting language was developed. In the previous sections it was explained that this method can be applied either if the attacker is accessing the network from a switch or from an access point and the steps for the implementation of the algorithm in both cases was described. However, for the purpose of this project, a script for detection of IP spoofing attacks on networks in which switches are the only Layer 2 devices was developed. Thus, if there is any AP present in the network, it will be defined as an unmanaged device.

In terms of the actual content of the script, it starts by requesting the SNMP version to the user as well as the community string or the username and authentication password, depending on the version in use. Then, information from previous executions of the script is deleted and managing IP addresses from all routers and switches present in the network are read from auxiliary text files and put in separate arrays. To start the detection process, function *Detect_IPSpoofing* is executed in an infinite loop.

This function will analyze sequentially each router contained on the routers' array, using a while loop. For each router, it starts by retrieving the device hostname from the *hostName* MIB object using a "snmpwalk" command combined with a "cut" command. Then, a *Router* function is executed to retrieve other necessary information from the MIB of the router. As it was said, this information is obtained from the *ipNetToMediaNetAddress*, *ipNetToMediaType* and *atPhysAddress* MIB objects using also the two previous commands and put in separate arrays. Now, each IP address from the corresponding array should be analyzed. To select only dynamic IP addresses from end hosts, an "if" statement is used to check the address type associated to the IP address.

As previously said, a record of all end hosts' IP addresses is kept. This information is

maintained in an auxiliary text file. After selecting a dynamic IP address corresponding to an end device, it is necessary to check if that IP address was already written into the file or not. This is done by reading each line of the file and comparing with the IP address. In case the IP address is not present in the register, it is added to the file. Also the MAC address associated to the IP address is written into a specific text file containing all MAC addresses from end hosts. The other parameter that must be saved is the registration time. Like it was done for the case of MAC spoofing attack detection, a "date" command is executed to obtain information about hours, minutes and seconds. Then, hours and minutes are converted into seconds and registration time is written into another text file as the total number of seconds. For the same reason mentioned in the previous chapter, it was also created a record for the registration date.

For cases in which an IP address of an end device was already recorded, the corresponding line number in the file is verified. Then, the same line is read from the file containing all MAC addresses. This allows consulting the MAC address associated to the IP address. If the present MAC address is the same as the recorded one, it means that the device possessing that IP address is still the same and the next IP address from the array is read. Otherwise, if the MAC addresses are different, it must be verified in first place if an IP spoofing attack has just been blocked. When a network attack of this type is blocked, the last information contained in the register is from the intruder's host, which means that it must be updated. The MAC address from the authorized client replaces the previous one in the file using the command "sed". Also the registration time and date is updated using the "date" command to obtain the information and the "sed" command to replace the old one. Besides, the variable signalizing the attack blocking is set to '0'.

When MAC addresses are different but any IP spoofing attack wasn't recently blocked, it is necessary to check how much times has passed since the MAC address was updated for the last time. First, the registration date is read from the corresponding file and from the line associated to the IP address. Then, it is compared to the present date. Again, as it was explained in the implementation of the MAC spoofing attack detection, there could be the possibility that an attack is performed on a day change, but since the two dates are different it is assumed that the change on the device associated to the IP address happened after a time period considered normal. Thus, if this happens, the "sed" command is executed to update the information in the files with the new device MAC address, date and time. On other hand, if the dates are different, it is necessary to compare the time elapsed since the information associated to the IP address was updated for the last time. So, the text file containing the registration times is analyzed. When the correct time is read, it is subtracted to the present time, which is obtained executing the "date" command and converted into seconds. The difference between both corresponds to the seconds that has passed.

As previously explained, once an IP address is configured on a machine, it is very likely that this IP address won't belong to any other device at least for a time period of large minutes. For the implementation of this algorithm, this time period was defined as 30 minutes, which corresponds to 1800 seconds. Thus, when the difference between the present time and the registration time is greater than this time period, it is considered that a new device was configured with the IP address and no network attack is present. In this situation, the MAC address from the new end device should replace the old one and the registration time is updated (since the date hasn't changed, it doesn't need to be updated). Again, "sed" command is used to perform this task. If the time passed is shorter than the defined 30 seconds, it is immediately assumed that the network is under an IP spoofing attack. The

MAC address of the intruder is the one that it is associated to the IP address at the present moment and a *Block_Spoofing* function is executed by the script to find this device in the network and block the attack.

To find the attacker's host, each switch is sequentially analyzed by reading their IP addresses from the array. Then, function *Switch* is executed to retrieve information from the switch MIB. The MIB objects *dot1dTpFdbAddress*, *dot1dTpFdbPort* and *atPhysAddress* are retrieved using the "snmpwalk" command combined with "cut" command and put in separate arrays, as usual. Then, it is executed a process over the array of bridge ports of the switch that it is similar to the one described in the implementation of the MAC spoofing detection algorithm. This process won't be described again, but it will basically select the access ports associated to end devices because these are the only ports where the intruder's host could be connected to. After the ports have been selected, it is performed a while cycle to analyze each one of them. For each selected port, the MAC address of the device connected there is read from the array. The MAC addresses are compared to the one that must be blocked until they match. When the intruder's MAC address is found, it is retrieved from the *dot1dBasePortIfIndex* MIB object the interface index associated to the bridge port using the "snmpget" command. It is also possible to retrieve the interface name using the *ifDescr* MIB object. Finally, to block the attack a "snmpset" command is executed over the *ifAdminStatus* MIB object to turn the interface down. To finish the process, a variable signalizing the attack block is set with value '1' so the information about the authorized client's host can be registered again.

In this section, we described an implementation of the metodologies for the detection and blocking of IP spoofing attacks, which were previously presented. In the following chapter, we will perform some tests in which this script as well as the previous ones will be executed over certain network scenarios in order to verify their efficiency and evaluate their performance.

Chapter 6

Network Equipment and Software

The previous chapters presented three different methodologies for network monitoring. The first one was directed for the discovery of all equipment present in the network and the other two were related to security issues, specifically to the detection of network attacks, in particular MAC spoofing and IP spoofing attacks. After the description of these algorithms, one possible implementation for each one of these methodologies was also presented. Now, it is necessary to validate the efficiency of the developed scripts and, in general, of all algorithms.

In order to test the previous methodologies, some network scenarios were created. Then, the developed scripts should be executed over these networks in order to obtain results. To facilitate the deployment of the networks, a network simulator software was used, where virtual equipment was connected. This allows to easily create networks with much more flexibility than using real equipment. However, it was also necessary to use some real devices to perform certain tasks, which were connected to the simulated equipment. This means that the created scenarios combine real devices and virtual equipment to form networks in which the previous methodologies should be deployed and tested.

This chapter starts by presenting the chosen network simulation software used on the deployment of networks for test purposes and then, all used equipment will be presented. For both virtual and real devices, we will present a description and show how the network equipment is configured. In particular, in case of real devices, it is also explained how they were connected to the simulated network.

6.1 Network Simulation Software

As it was mentioned, using exclusively real equipment for the creation of testing networks is not reasonable. The deployment of real networks is not practical and most of all, it can be really expensive depending on the size of the network and the number of computers, routers, switches or other network devices to be used. Nowadays, several softwares that are able to perform network simulation have been developed and many of them can recreate exactly any detail of a real network. This permits to easily deploy a network with the benefit of saving space, money and time. Other point to have in consideration is that SNMP retrieves information from the MIB of each network device but depending on the device manufacturer, the MIB objects can be organized in different ways. Since the implementation of the previous methodologies were developed for Cisco network devices, which are probably the most used, it makes sense to look for some network simulation programs that emulate real Cisco hardware

and software and check which one would better fit the purpose of this project. So, three different programs were taken into consideration for the choice of the network simulator software: Cisco Packet Tracer, NetSim and GNS3.

6.1.1 Cisco Packet Tracer

Cisco Packet Tracer is a network simulation software developed by Cisco for educational purposes, in order to support its Networking Academy program. This software provides visual simulation of networks and allows the creation of networks with an almost unlimited number of devices. The configuration of the network devices is done through a command-line interface just like in real equipment. It also provides tables, diagrams and other visual representation and offers a multiuser functionality that permits multiple users to work on the same project through the Internet [68]. This software has two operation modes available to analyze the network behavior. The first is the real-time mode that shows how real devices would behave and the immediate network response to any network change; the other is the simulation mode and it is directed to background concepts and allows to control time intervals, data transfer rates, bandwidth and manipulate the propagation of data packets through the network. This software supports most of network protocols, including SNMP, which is the basis of the developed algorithms. Some other characteristics of this software are the possibility of inserting interface cards into modular routers and switches, creating virtual networks over real ones and its compatibility with Windows and Linux (Ubuntu and Fedora) operating systems. On a first view, this network simulator program have all the characteristics needed for the purpose of this project, but the fact that it is only available for instructors, students and alumni registered on the Cisco Networking Academy makes the choice of this software not possible [69].

6.1.2 Boson NetSim - Network Simulator

Other network simulator software is NetSim, which is a Cisco network simulator developed by Boson. Boson is a company that provides material to prepare students for IT certification exams from Cisco, Microsoft CompTIA and others. NetSim is one of the tools developed to prepare users for Cisco certification exams [70]. It uses Network Simulator, Router Simulator and EROUTER software technologies to simulate a real network and it is available on three different versions, each one with specific characteristics and directed for a different Cisco certification. Focusing on its features, NetSim supports up to 42 routers and 6 switches on a total of 200 devices on the network, it simulates network traffic with virtual packet technology and it provides Telnet mode or Console mode to interact with the network devices [70]. Although it supports Telnet, SNMP was the chosen remote monitoring protocol and unfortunately this software doesn't support it. Furthermore, this software is not a free application and that is one of the priorities on the software's choice.

6.1.3 GNS3: Graphical Network Simulator

Finally, the last software is called GNS3. GNS3 is a graphical network simulator that emulates complex networks and is used mainly by network engineers, administrators and students to prepare themselves for Cisco and Juniper certification exams. This network software takes advantage from other programs to turn simulations more similar to real labs. The first program is Dynamips, which is the core program that allows running and emulating

a Cisco IOS in a virtual environment. GNS3 works as the GUI part that runs over Dynamips to provide a graphical interface. This way, a user can easily create different network topologies using a diversity of Cisco routers. While Dynamips is responsible for the back-end operation of emulating routers with real IOS images, GNS3 uses Dynagen as the text-based front-end to establish communication with Dynamips [71]. GNS3 also supports other machine emulators and virtualizers like Qemu, Virtualbox or Pemu. This allows a user to simulate networks with a wide diversity of devices like Cisco ASA and PIX firewalls, Cisco IPS, Juniper routers or hosts (Linux, Windows, MacOS X, etc.). Other advantages of GNS3 are the possibility of connecting the virtual network to the real world with real devices and performing packet capture using Wireshark. Switching is also possible to emulate by using an EtherSwitch card in a router. Finally, in terms of features, GNS3 is an open source software that may be used in operating systems like Windows, Linux and MacOS X. Some limitations of GNS3 are its limited throughput, which is 1000 packets per second in the virtual environment (in a real router it would provide a much greater throughput) and the large amount of real and virtual memory that it can consume and CPU usage [72]. About this last point, GNS3 is already prepared with some tools in order to prevent this memory and CPU usage issues but it must be always taken into account that the higher the number of routers and network devices, the higher will be the consume [73].

After analyzing the characteristics of all this software, GNS3 was selected as the network emulator software tha best fits the purpose of this dissertation. It's a really complete tool where actual Cisco IOS are being emulated with all characteristics of real Cisco devices. The fact that Cisco IOS supports SNMP protocol and the possibility to combine virtual devices to real equipment turns GNS3 into the perfect tool to create testing scenarios for the previous methodologies.

6.2 Virtual Equipment

Now that the network simulation software was chosen, it will be presented the devices to be simulated in this program. GNS3/Dynamips is a powerful tool, capable to simulate Cisco routers by running real Cisco IOS. So, instead of using real routers with all already mentioned disadvantages, virtual Cisco routers are used, which GNS3 emulates with the same exact behavior of real ones.

First, it is necessary to get the desired Cisco IOS image and add it to the list of IOS images in GNS3 to be able to deploy it in any network. For the purpose of this project, we used Cisco routers using an IOS image from a Cisco C3640 router. Each of these routers consumes 128Mb of RAM and it can support one FastEthernet interface, up to five Ethernet interfaces and an EtherSwitch card with up to sixteen ports. When running a router for the first time, it is necessary to define an Idle PC value. This is a functionality provided by GNS3 to reduce the CPU usage, allowing the addiction of more network devices. The configuration of the routers is done through a command-line interface. Following, Cisco commands representing the configurations made on each router to assure that they would be correctly deployed on the network are shown, assuring that the developed scripts would be successfully tested.

First, it is necessary to assign the router with a hostname. The hostname will allow to identify each network device and it is essential for the correct execution of the developed algorithms:

Router>enable
Router# config t
Router(config)# hostname [Router Hostname]

Then, it is necessary to configure each used interface with an IP address and also enable it, in order to connect the router with other devices. Normally, if a router interface is connected to other Layer 3 devices or to a Layer 2 device in a network configured with a single VLAN, the interfaces configuration is as follow:

Hostname(config)# interface [Interface Name]
Hostname(config-if)# ip address [IP Address] [Network Mask]
Hostname(config-if)# no shutdown

For networks with more than one VLAN, it is needed a router to route the traffic and allow network devices in different VLANs to communicate with each other [74]. So, in the router interface directly connected to this network, we should create sub interfaces associated to each VLAN. This method is called Router-on-a-stick and allows different VLANs to communicate via the sub interfaces of the router. In this situation, the creation of sub interfaces over an interface is as follows:

Hostname(config)# interface [Interface Name]
Hostname(config-if)# no shutdown
Hostname(config-if)# interface [Interface Name].[Sub Interface Number]
Hostname(config-if)# encapsulation dot1Q [Vlan Number]
Hostname(config-if)# ip address [IP Address] [Network Mask]

To correctly route packets over the whole network it is necessary to enable a routing protocol. The RIP protocol was the chosen one for test purposes. It is a distance-vector routing protocol in which an algorithm is used to calculate paths based on the information sent periodically from neighbor routers [75].

Hostname(config)# router rip
Hostname(config-router)# network [Directly Connected Network]

The last command must be executed for each network that is directly connected to the router. Obviously, it is also necessary to configure the router as an SNMP server so the local machine can send SNMP commands to retrieve information from its MIB. The following commands enable SNMP version 2 and version 3, depending on the desired one:

- SNMP Version 2:

Hostname(config)# snmp-server community [Community String] RW

- SNMP Version 3:

Hostname(config)# snmp-server group [Groupname] v3 auth
Hostname(config)# snmp-server user [Username] [Groupname] v3 auth md5 [Password]

Finally, to avoid the configuration of each router every time a network simulation starts, it is possible to copy the current configuration to a TFTP server that should be configured on the local computer. Of course, this is only possible after the communication between router and local host is working properly. Then, this configuration file should be set as the startup configuration when simulation starts. This is defined in GNS3 for each router so every time it is initialized, it will upload the file and the router will be immediately connected and ready to use. The copy command that should be executed after the previous configurations have been conducted is:

Hostname# copy running-config tftp

The previous commands provide routers with the basic configuration, necessary for testing purposes. Of course, other router parameters can be defined and configured according to the network characteristics.

Routers using an EtherSwitch card to work exclusively as a switch have been defined as unmanaged devices for the network discovery algorithm presented earlier. However, these devices can still exist on the network and they should actually be deployed for test purposes to observe if the developed script can actually ignore them. The configuration of routers using switching modules should be the same as any normal switch and it will be described in the next section when the configuration of real switches is discussed.

To simulate real network scenarios, it is also necessary to have many end hosts communicating over the network. As previously referred, GNS3 provides the possibility of emulating other devices using virtualizers such as VirtualBox. After VirtualBox has been correctly configured on GNS3, it is necessary to choose an operating system for the emulation of virtual hosts. For experimental tests, only simple operations need to be executed from these hosts, so Linux Microcore 4.0.2 was chosen as the operating system to run on end devices. Since there's no need for many features, Linux Microcore is a minimal operating system with only 8Mb size and a command-line interface that provides some basic Linux commands. To connect each virtual host to the rest of the network it should be configured with the following commands:

$ sudo ifconfig eth1 [Host IP Address] netmask [Network Mask] up
$ sudo route add default gw [Default Gateway] eth1

The first command will assign the interface Ethernet1 of the virtual host with an IP address. Ethernet1 is the default interface to establish the communication between the emulated host and GNS3. The second command will define the default gateway associated to this host. If all parameters are correctly introduced, the virtual machine is now able to communicate with the network. As in case of routers, this was the configuration made for test purposes.

In terms of virtual equipment, Dynamips and VirtualBox emulate, respectively, routers and end hosts on the local machine and they can be easily deployed on any virtual network created on GNS3 by simply adding the correspondent device symbol to the workspace. The described device configurations allowed the simulation of networks and a posterior execution of the developed scripts. In the next section, we will present the physical equipment used on the implementation of networks.

6.3 Real Equipment

We have just mentioned the advantages of using virtual devices. The main reasons are its flexibility and the possibility of easily creating networks containing more devices. However, some of the devices used for network simulations on this project were real instead of emulated by some application. The first one is the local computer that works as the monitoring station. It would be expected that this computer was real and not emulated as any of the other end devices for the simple reason that all operations are executed from this machine, including the emulation of all the previous virtual devices and the execution of the developed scripts in which SNMP packets are exchanged between this computer (the manager) and all other network devices (the agents). As mentioned in a previous chapter, the operating system used on the local computer is Ubuntu. To integrate this machine in the network in order to be able to communicate with all other network devices, it is necessary to configure it. The following network configuration was deployed in practice and it represents the basic configuration that allows the connection of the computer to the network. Of course, other parameters can be configured according to the network where the host will be connected to.

$ sudo modprobe tun
$ sudo tunctl
$ sudo ifconfig tap0 [IP Address] netmask [Network Mask] up
$ sudo route add net [Network Address] netmask [Network Mask] gw [Default Gateway] dev tap0

To establish the communication between the local computer and the virtual network running on GNS3, it is necessary to use a loopback interface. Obviously, this is not a real interface but a virtual network interface that allows the communication between network applications running on the same machine [76]. In Ubuntu, the loopback interface is called a tap interface. The first two commands initialize this interface, so it can be used normally as any other physical interface. In particular, the first command uploads the tun module, necessary for the use of the tap interface which is created by the second command. If this second command is executed more than once, more tap interfaces will be created and made available for use. Then, the next command assigns the created loopback interface with an IP address and turns the interface on. Finally, the last command is necessary to define a default gateway for each network and it should be executed for all virtual networks. This way, each packet sent from the tap interface to any network will be directed to the corresponding router. In this case, we didn't executed a simple command to define a unique default gateway, like it was done for virtual hosts, because the local computer could have other physical interfaces connected, for example, to the Internet and the packets sent from these interfaces would also be directed to this router which wouldn't know where to route them. After the computer have been configured, it is necessary to add it to the virtual network in GNS3. When using real devices, a cloud symbol is used on GNS3 workspace to establish the connection. Then, on the cloud configurations, the *NIO TAP* tab is selected, in which the interface used by the local machine (tap0, in the case) is introduced. The monitoring station is now ready to be connected to the virtual network created on GNS3.

Another physical device used on the development of network scenarios was a Switch. GNS3 is a very complete software with most of the necessary features for a network simulation, but unfortunately Dynamips is not able to emulate Cisco switches (or from any other

manufacturer) like it does for routers. Actually, it provides some alternative solutions, like using simple Ethernet Switch devices or introducing EtherSwitch cards into routers and using it only for switching tasks [77]. The first solution is not viable because it provides such a simple simulation of a switch that it isn't even possible to configure it or interact with it. It only allows defining the port type (access or trunk) and VLANs. The other possibility come with an already mentioned problem. Routers using EtherSwitch cards can perform most of the tasks that a normal switch does and for the purpose of this project it would work perfectly, but its MIB contain a lack of information essential for the correct application of the developed algorithms. Without the missing MIB objects, it is not possible to test the scripts and that was the reason why this couldn't be considered an option for switching tasks. So, the best solution is to use a real switch with all its Layer 2 functionalities and a complete MIB with all the necessary MIB objects. The real switch used in practice for the creation of networks was a Cisco Catalyst C3750. As any other network device, it should be correctly configured so it can be connected to the network and provide all switching features. Following, we will describe the configuration implemented in practice.

First, it is necessary to define a device hostname so it can be identified and distinguished from other equipment:

Switch>enable
Switch# config t
Switch(config)# hostname [Switch Hostname]

Then, if there is more than a single VLAN on the network, each of them should be defined as follows:

Hostname# vlan database
Hostname(vlan)# vlan [Vlan Number]

Now, the essential part of the configuration process is to define the switch ports. Each interface used should be configured as an access port or trunk port to be connected, respectively, to end devices and other network devices or exclusively to other network devices like routers and switches.

To define a switch interface as an access port, we should execute the commands:

Hostname(config)# interface [Interface Name]
Hostname(config-if)# switchport mode access
Hostname(config-if)# switchport access vlan [Vlan Number]
Hostname(config-if)# no shutdown

On the other hand, to define an interface as a trunk port, the commands are:

Hostname(config)# interface [Interface Name]
Hostname(config-if)# switchport mode trunk
Hostname(config-if)# switchport trunk encapsulation dot1q
Hostname(config-if)# no shutdown

Since SNMP is used to retrieve information from the network devices, it is necessary that each one is assigned with an IP address, so the local host can access its MIB. Thus, an IP address must be configured on the switch only for managing purposes:

Hostname(config)# interface Vlan [Vlan Number]
Hostname(config-if)# ip address [IP Address] [Network Mask]

Finally, the switch should be defined as an SNMP server to allow the exchange of SNMP packets between the device and the manager (the local computer). As in case of routers, the commands for the configuration of an SNMP server using version 2 and version 3 are:

- SNMP Version 2:

Hostname(config)#snmp-server community [Community String] RW

- SNMP Version 3:

Hostname(config)#snmp-server group [Groupname] v3 auth
Hostname(config)#snmp-server user [Username] [Groupname] v3 auth md5 [Password]

Finally, to avoid repeating all the previous steps each time a simulation is executed and the switch is started, this configuration is defined as the startup configuration of the switch.

All the previous commands provide the switch with the basic configuration to be connected to the network. These configurations can also be applied to emulated routers using EtherSwitch cards. Since this is a physical device, it must be connected to GNS3 to communicate with the rest of the network. To do so, first the switch must be connected to one of the Ethernet interfaces of the local computer. Then, in GNS3 is added a cloud symbol to the workspace which will represent the switch. On the cloud configuration page, the *NIO Ethernet* tab is selected and the Ethernet interface name of the local PC where the switch is connected to is added. The switch is now ready to be connected to the network and this is a simple procedure to associate a real switch (or any other real network device) to a virtual network.

With the usage of real switches comes another situation. One of the switch ports was connected to a local host interface to establish the communication with a virtual network running emulated devices on GNS3. But since this is a physical device with real interfaces, only real equipment can be connected there. The configuration of all network devices present in the network simulations for test purposes was already described, so any real device connected to the switch should be configured as previously explained for the corresponding device type. To test the developed algorithms, real end hosts were connected to the switch. These hosts were simple laptops running the Ubuntu operating system. Thus, the network configuration of these devices is the same as described in the previous section for virtual hosts.

All network equipment, physical and virtual, used on network simulations as well as its network configurations have been described. In the next section, we will present the network scenario created to test each of the developed scripts.

Chapter 7

Experimental Results

After presenting all network equipment and network simulation software used to perform the necessary experimental tests, it is now time to describe the network scenario that was created to test the efficiency of the developed methodologies in different situations. This simulated network will try to embrace a whole set of conditions in order to prepare the algorithms to be deployed in a real network. Then, all the results obtained from the execution of the developed scripts over the simulated network will be analyzed. These tests will allow us to validate the network discovery, IP spoofing and MAC spoofing attacks detection algorithms and evaluate the performance of these methodologies.

7.1 Testing Scenario

In order to perform the simulation tests, the scenario presented on Fig. 7.1 was created. The backbone of this network is composed by four Cisco C3640 routers connected to each other, forming a mesh. This topology intends to test the efficiency of the network discovery algorithm; as previously said, when performing a network discovery, the algorithm has to make sure that all managed devices are analyzed once and this network topology intends to assure that this actually happens whatever the network is. Another Cisco C3640 router (*R5*) is connected to router *R4* to test the stopping network feature. This functionality intends to establish a border between what should and shouldn't be discovered and, therefore, between what should be monitored and what should remain private. Thus, by defining the network that establishes the connection between these two routers (*17.1.1.0*) as the stopping network, it will be possible to check if the algorithm actually stops and verify if it doesn't discover *R5*. If it works, then we can be sure that it wouldn't also search for any other network equipment beyond router *R5*.

Then, the device identified as *PC* represents the local machine that it will work as the monitoring station to manage the network. Router *R1* and *PC* are connected to a router Cisco C3725 using a switching module (*SWR1*), which in practice is considered an unmanaged device due to its incomplete MIB but it can be identified by the algorithm. *PC1* is a virtual host running Linux Microcore operating system and it is also connected to the EtherSwitch router *SWR1*. *PC* was configured to belong to VLAN1 and *PC1* to VLAN 2. So, the *R5* router interface connected to this LAN was configured with sub-interfaces to provide communication between the two VLANs.

There is a network connected to router *R2* composed by two Cisco C3725 routers using

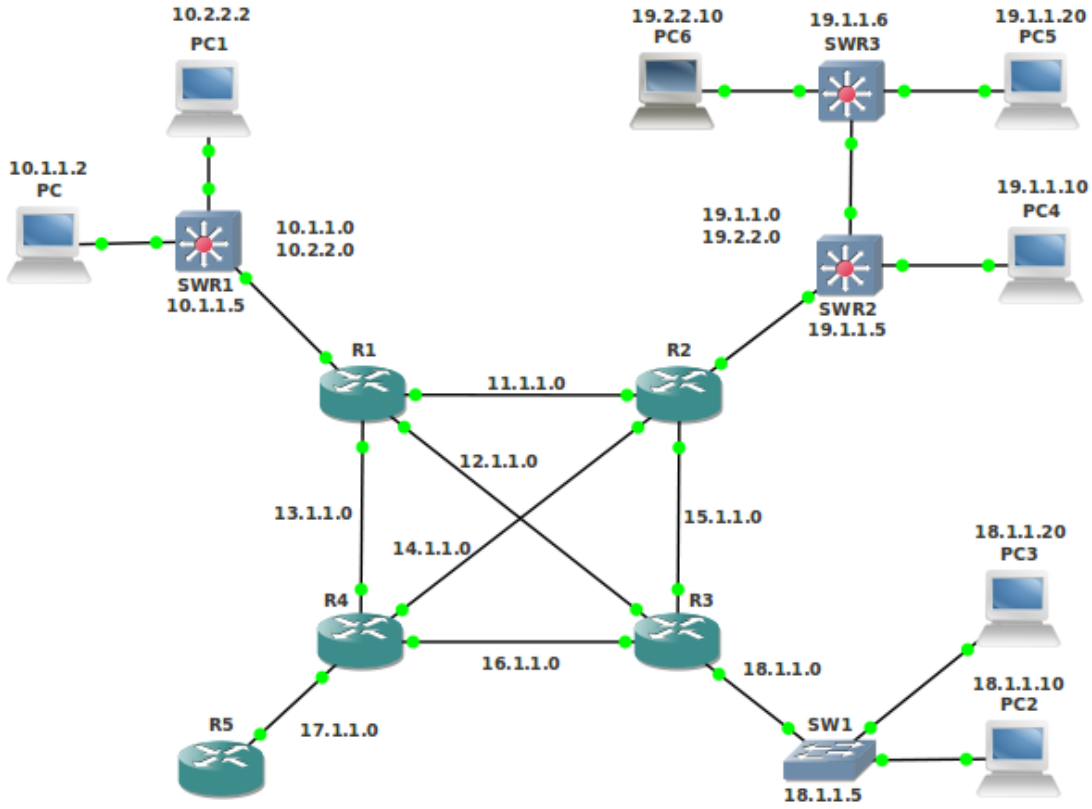


Figure 7.1: Simulated Network

an EtherSwitch card (*SWR2* and *SWR3*) and three virtual end hosts (*PC4*, *PC5* and *PC6*) running Linux Microcore operating systems. *PC4* and *PC5* belong to VLAN1 and they are connected to access ports of *SWR2* and *SWR3*, respectively. Then, *PC6* is connected to *SWR3* and belongs to VLAN2. The ports that connect *SWR2* and *SWR3* are defined as trunk ports, as well as the port that connect *SWR2* to *R2*. The *R2* interface connected to this LAN was configured with sub-interfaces in order to provide communication between the different VLANs.

Finally, the vitual router *R3* is connected to a real Cisco Catalyst C3750 switch (*SW1*), which is also connected to two real end hosts (*PC2* and *PC3*) running Ubuntu operating systems. These two hosts belong to the same VLAN. As it was said, the developed methodologies for the detection of spoofing attacks are only prepared for detection of attacks performed by hosts connected to the network through switches. In the network scenario from Fig.7.1, the unique real switch is *SW1*. Thus, beyond the discovery of this switch, this LAN intends to test and evaluate the efficiency of both MAC spoofing and IP spoofing detection methodologies. The two hosts will be used to simulate MAC spoofing and IP spoofing attacks by simulating a user with an authorized access to the network and an intruder that it will impersonate this user to get access to the network.

Before proceeding to the next section, in which the experimental tests will be performed and the obtained results will be analyzed, Table 7.1 shows some information related to each end host present in the network. This information includes the device hostname, MAC address

and IP address and it intends to facilitate the task of analyzing the information displayed after running the network discovery algorithm in the next section.

Table 7.1: End Hosts Information

Hostname	MAC Address	IP Address
PC	56 AD 66 B4 B2 47	10.1.1.2
PC1	08 00 27 07 49 3D	10.2.2.2
PC2	00 13 D4 2E 5D 93	18.1.1.10
PC3	00 11 2F BB BB 44	18.1.1.20
PC4	08 00 27 23 1C 3E	19.1.1.10
PC5	08 00 27 6F 91 9D	19.1.1.20
PC6	08 00 27 8E 07 31	19.2.2.10

The whole network scenario was explained, as well as the reasons for such configuration and topology. Besides, information about each end host have also been presented, and finally the developed algorithms are ready to be executed by the local computer. In the following section, we will analyze the obtained results.

7.2 Analysis of Results

Starting from the network discovery algorithm, when this script is executed some information is required. Figures 7.2, 7.3 and 7.4 show three examples of requested information according to what the user introduces. First, it is necessary to provide the IP address of any router in the network. It is irrelevant which router is selected because the algorithm is developed in order to discover all network devices, no matter what the first router is. For test purposes it was introduced the IP address from one of router *R1* interfaces (*10.1.1.1*). Then, it is asked if the user wants to discover the whole network or not. In case of a negative answer (Fig. 7.3), the user should insert the first network that won't be analyzed by the algorithm and, consequently, none of the following networks will be analyzed. Otherwise (Fig. 7.2), there will be no limitations and every device on any network will be analyzed. Then, MySQL database systems require account information from the user to establish the connection to the server. Once the connection is established, it is possible to create and manage databases. So, the next step is to introduce the username and password from a MySQL account that the user has already created.

Finally, SNMP information is needed. SNMP provides the remote interaction between the local machine and the network equipment. The local computer will work as the manager or client, while each network device works as an agent, which means that they must be configured as a SNMP server. The algorithm was developed in order to work with both versions 2 and 3 of SNMP. This gives the user the freedom to choose the version that fits better on his interests. So, the user should insert the SNMP version that was configured on the network equipment to provide compatibility with the SNMP commands executed from the local computer. If the user chooses the version 2 of SNMP (Fig. 7.2), then the community string configured on the devices should be introduced. Otherwise, in case of version 3 (Fig. 7.4), the username and authentication password are required.

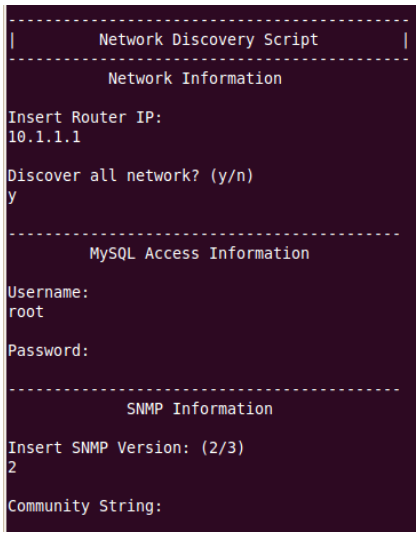


Figure 7.2: Information Requested - SNMP v2 w/o Stopping Network

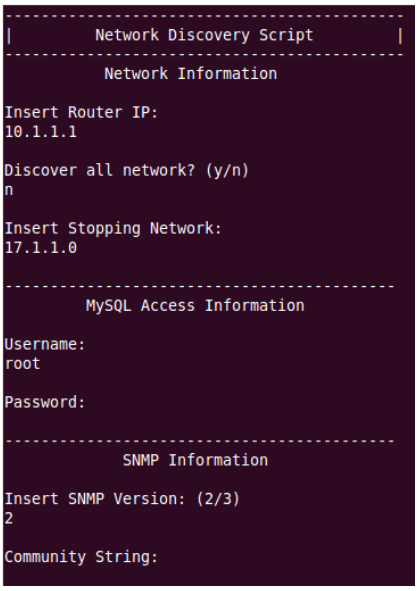


Figure 7.3: Information Requested - SNMP v2 w/ Stopping Network

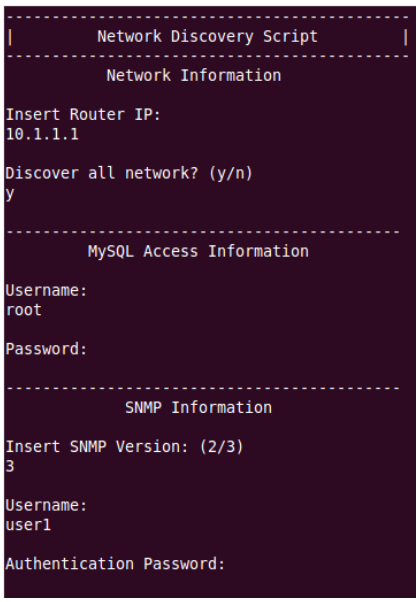


Figure 7.4: Information Requested - SNMP v3 w/o Stopping Network

After all previous information has been requested and introduced by the user, the algorithm have all the necessary information to perform network discovery. For the network scenario from Fig. 7.1 the algorithm took 3 minutes and 15.741 seconds since it was executed until it finished the whole discovery process. This time value was obtained using the *time* command, which returns the exact time that a process takes to be executed. When the algorithm execution finally stops, it is possible to consult the output information that is displayed, i.e, the information obtained from all network devices that support SNMP. Starting from router *R1*, the output information is presented in Fig. 7.5.

As can be seen, information retrieved from the router MIB was carefully selected and organized in a readable way. This allowed to obtain the actual Routing table and ARP table that would be seen if the respective commands were executed on the command-line interface of the router. In the top of the image, the device type, hostname, manufacturer and model are described. The last section of this figure presents a table with the IP addresses assigned to each router interface. By sending only SNMP commands from the local computer, it was possible to obtain these three tables and the device information, which were the main objectives for this type of equipment. Now it is necessary to analyze the output information from the other routers present in the network.

```
--------------------------------------------------------------------------
| Router R1 - Cisco C3640                                                |
--------------------------------------------------------------------------
 ROUTING TABLE

 C   10.1.1.0/24 is Directly Connected, FastEthernet0/0.1
 C   10.2.2.0/24 is Directly Connected, FastEthernet0/0.2
 C   10.3.3.0/24 is Directly Connected, FastEthernet0/0.3
 C   11.1.1.0/24 is Directly Connected, Ethernet1/1
 C   12.1.1.0/24 is Directly Connected, Ethernet1/2
 C   13.1.1.0/24 is Directly Connected, Ethernet1/0
 R   14.0.0.0/8 [1] via 11.1.1.2, Ethernet1/1
                [1] via 13.1.1.4, Ethernet1/0
 R   15.0.0.0/8 [1] via 11.1.1.2, Ethernet1/1
                [1] via 12.1.1.3, Ethernet1/2
 R   16.0.0.0/8 [1] via 12.1.1.3, Ethernet1/2
                [1] via 13.1.1.4, Ethernet1/0
 R   17.0.0.0/8 [1] via 13.1.1.4, Ethernet1/0
 R   18.0.0.0/8 [1] via 12.1.1.3, Ethernet1/2
 R   19.0.0.0/8 [1] via 11.1.1.2, Ethernet1/1
--------------------------------------------------------------------------
  ARP TABLE

IP Address        MAC Address        Address Type       Interface
 13.1.1.1      CC 04 0C 3F 00 10       static          Ethernet1/0
 13.1.1.4      CC 03 0C 3F 00 10       dynamic         Ethernet1/0
 11.1.1.1      CC 04 0C 3F 00 11       static          Ethernet1/1
 11.1.1.2      CC 05 0C 3F 00 11       dynamic         Ethernet1/1
 12.1.1.1      CC 04 0C 3F 00 12       static          Ethernet1/2
 12.1.1.3      CC 06 0C 3F 00 12       dynamic         Ethernet1/2
 10.1.1.1      CC 04 0C 3F 00 00       static          FastEthernet0/0.1
 10.1.1.2      56 AD 66 B4 B2 47       dynamic         FastEthernet0/0.1
 10.1.1.5      C2 02 0C 30 00 00       dynamic         FastEthernet0/0.1
 10.2.2.1      CC 04 0C 3F 00 00       static          FastEthernet0/0.2
 10.2.2.2      08 00 27 07 49 3D       dynamic         FastEthernet0/0.2
 10.3.3.1      CC 04 0C 3F 00 00       static          FastEthernet0/0.3
--------------------------------------------------------------------------
  IP ADDRESS TABLE

IP Address         Interface
 10.1.1.1         FastEthernet0/0.1
 10.2.2.1         FastEthernet0/0.2
 10.3.3.1         FastEthernet0/0.3
 11.1.1.1         Ethernet1/1
 12.1.1.1         Ethernet1/2
 13.1.1.1         Ethernet1/0
--------------------------------------------------------------------------
```

Figure 7.5: Router R1 Output

The information obtained from routers *R2*, *R3*, *R4* and *R5* are present in Figures 7.6, 7.7, 7.8 and 7.9, respectively.

```
| Router R2 - Cisco C3640                                                    |
ROUTING TABLE

R   10.0.0.0/8 [1] via 11.1.1.1, Ethernet1/1
C   11.1.1.0/24 is Directly Connected, Ethernet1/1
R   12.0.0.0/8 [1] via 11.1.1.1, Ethernet1/1
               [1] via 15.1.1.3, Ethernet1/0
R   13.0.0.0/8 [1] via 11.1.1.1, Ethernet1/1
               [1] via 14.1.1.4, Ethernet1/2
C   14.1.1.0/24 is Directly Connected, Ethernet1/2
C   15.1.1.0/24 is Directly Connected, Ethernet1/0
R   16.0.0.0/8 [1] via 14.1.1.4, Ethernet1/2
               [1] via 15.1.1.3, Ethernet1/0
R   17.0.0.0/8 [1] via 14.1.1.4, Ethernet1/2
R   18.0.0.0/8 [1] via 15.1.1.3, Ethernet1/0
C   19.1.1.0/24 is Directly Connected, FastEthernet0/0.1
C   19.2.2.0/24 is Directly Connected, FastEthernet0/0.2
C   19.3.3.0/24 is Directly Connected, FastEthernet0/0.3
------------------------------------------------------------------------------
 ARP TABLE

IP Address        MAC Address        Address Type      Interface
 15.1.1.2      CC 05 0C 3F 00 10       static          Ethernet1/0
 15.1.1.3      CC 06 0C 3F 00 10       dynamic         Ethernet1/0
 11.1.1.1      CC 04 0C 3F 00 11       dynamic         Ethernet1/1
 11.1.1.2      CC 05 0C 3F 00 11       static          Ethernet1/1
 14.1.1.2      CC 05 0C 3F 00 12       static          Ethernet1/2
 14.1.1.4      CC 03 0C 3F 00 12       dynamic         Ethernet1/2
 19.1.1.2      CC 05 0C 3F 00 00       static          FastEthernet0/0.1
 19.1.1.5      C2 00 0C 21 00 00       dynamic         FastEthernet0/0.1
 19.1.1.6      C2 01 0C 21 00 00       dynamic         FastEthernet0/0.1
 19.1.1.10      08 00 27 23 1C 3E       dynamic         FastEthernet0/0.1
 19.1.1.20      08 00 27 6F 91 9D       dynamic         FastEthernet0/0.1
 19.2.2.2      CC 05 0C 3F 00 00       static          FastEthernet0/0.2
 19.2.2.5      C2 00 0C 21 00 00       dynamic         FastEthernet0/0.2
 19.2.2.6      C2 01 0C 21 00 00       dynamic         FastEthernet0/0.2
 19.2.2.10      08 00 27 8E 07 31       dynamic         FastEthernet0/0.2
 19.3.3.2      CC 05 0C 3F 00 00       static          FastEthernet0/0.3
------------------------------------------------------------------------------
  IP ADDRESS TABLE

IP Address         Interface
 11.1.1.2         Ethernet1/1
 14.1.1.2         Ethernet1/2
 15.1.1.2         Ethernet1/0
 19.1.1.2         FastEthernet0/0.1
 19.2.2.2         FastEthernet0/0.2
 19.3.3.2         FastEthernet0/0.3
------------------------------------------------------------------------------
```

Figure 7.6: Router R2 Output

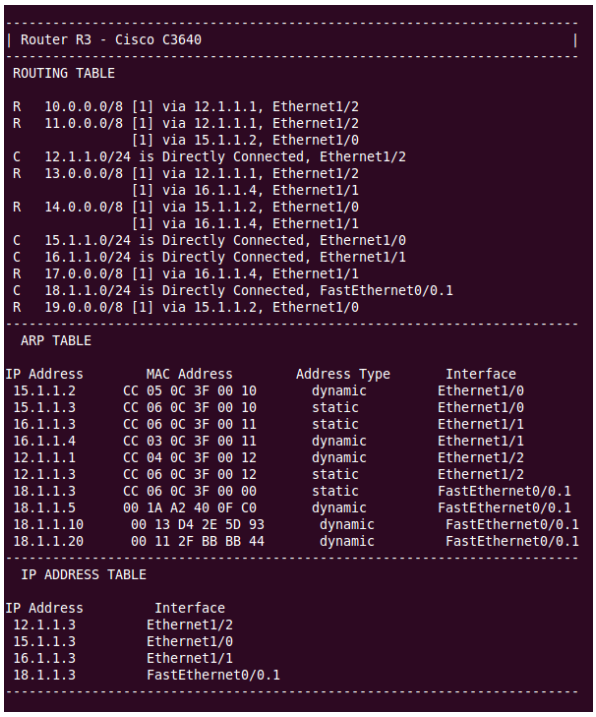


Figure 7.7: Router R3 Output

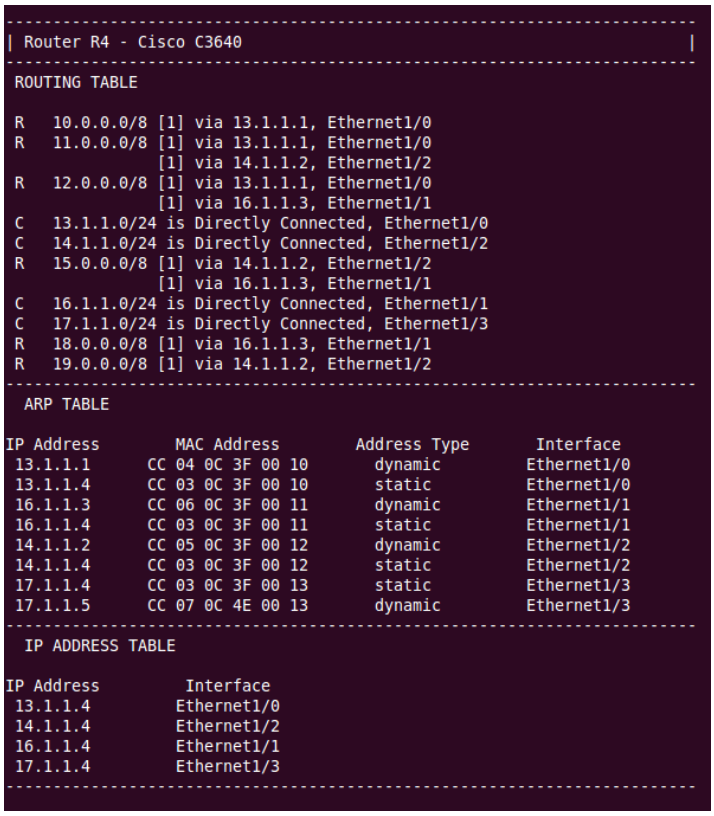


Figure 7.8: Router R4 Output

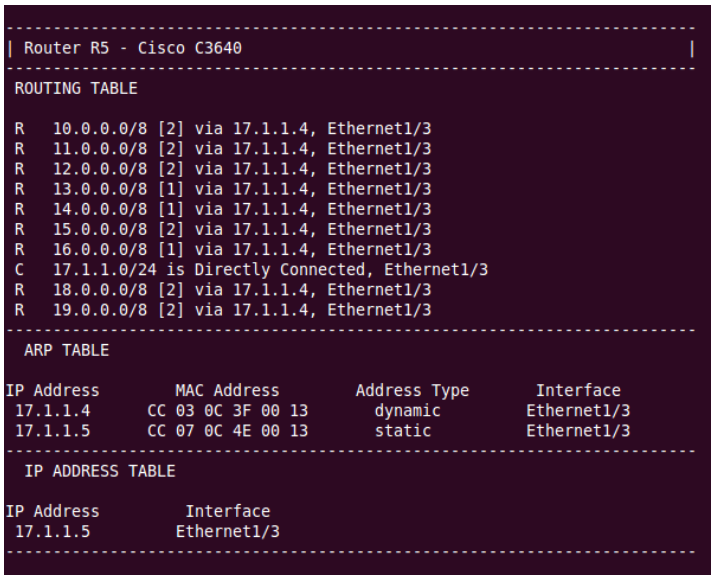


Figure 7.9: Router R5 Output

In all routers, we displayed the device information, Routing table, ARP table and IP addresses assigned to their interfaces. By analyzing Table 7.1 and the tables with the IP addresses from each router interface, and crossing this information with Routing tables and ARP tables, it can be concluded that the presented information is definitely correct.

Referring now to the real switch (*SW1*), the output information is shown in Fig. 7.10. In the top of the figure, the device information is described: device type, hostname, manufacturer and device model. Everything seems to be in accordance to the real information of the switch. Then, its MAC Address Table is displayed. Again, in order to create this table as shown, a variety of information from the switch MIB was selected and organized in a readable way in order to be presented to users. The three MAC addresses present in the table correspond to router *R3* and end hosts *PC2* and *PC3*. This information appears to be correct, which means that the algorithm can correctly analyze any real switch.

```
------------------------------------------------------------------------
| Switch SW1 - Cisco C3750ME                                           |
------------------------------------------------------------------------
  MAC ADDRESS TABLE

    MAC ADDRESS        Type          Interface          Port     VLAN
 00 11 2F BB BB 44    dynamic     FastEthernet1/0/2     Access     1
 00 13 D4 2E 5D 93    dynamic     FastEthernet1/0/3     Access     1
 CC 06 0C 3F 00 00    dynamic     FastEthernet1/0/1     Access     1
------------------------------------------------------------------------
```

Figure 7.10: Switch SW1 Output

In terms of routers using switching modules, it was explained that this type of devices cannot be discovered as any other switch. However, the algorithm is able to identify such devices. Figure 7.11 presents the information displayed by the algorithm when devices *SWR1*, *SWR2* and *SWR3* are detected. As it can be seen, in this case the algorithm doesn't display the MAC Address Table of the device, instead, it presents the device information (device type, hostname, manufacturer and model).

```
------------------------------------------------------------------------
| Router SWR1 - Cisco C3725 (w/ EtherSwitch card)                      |
------------------------------------------------------------------------

------------------------------------------------------------------------
| Router SWR2 - Cisco C3725 (w/ EtherSwitch card)                      |
------------------------------------------------------------------------

------------------------------------------------------------------------
| Router SWR3 - Cisco C3725 (w/ EtherSwitch card)                      |
------------------------------------------------------------------------
```

Figure 7.11: Routers SWR1, SWR2 and SWR3 Output

To confirm that the algorithm analyzes only SNMP-enabled devices, Fig. 7.12 shows the reaction of the algorithm when the end hosts present in the network, which don't support SNMP, are detected. Since the algorithm detects the presence of these devices, it is unavoidable that it tries to analyze them. So, when the local computer sends an SNMP command to any end host, a *timeout* message is returned meaning that there was no reply from the device. Then, the algorithm simply ignores this host and proceeds to next IP address from the list.

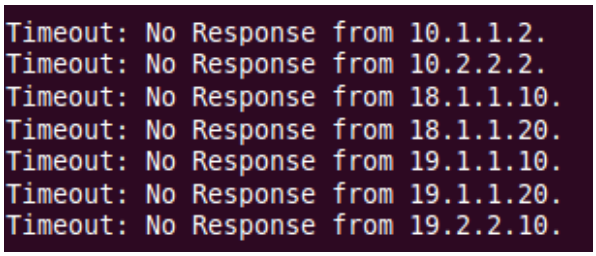


Figure 7.12: End Hosts Output

During the algorithm description, it was referred that this algorithm would also discover connections between the different routers. This is actually the last procedure, after all network devices have been discovered. In Fig. 7.13 it can be verified the routers' connections displayed by the algorithm. Comparing it with the simulated network from Fig. 7.1, it can be concluded that all connections between routers have been discovered.

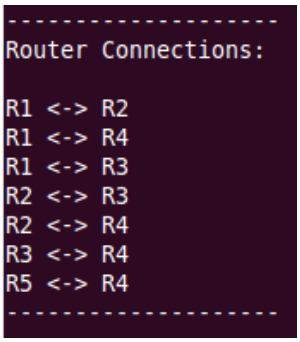


Figure 7.13: Router Connections Output

Finally, the last objective of this algorithm was to record all this gathered information into a database system. phpMyAdmin was the management tool used to administrate the MySQL database that was created. Fig. 7.14 shows that the database tables were actually created after the algorithm have been executed. As it can be seen, five tables were created: one with information from all the SNMP-enabled devices found in the network and the other four with information from Routing Tables, ARP Tables, IP Addresses from all routers' interfaces and MAC Address Tables. These tables will allow to record all the information displayed by the network discovery algorithm.

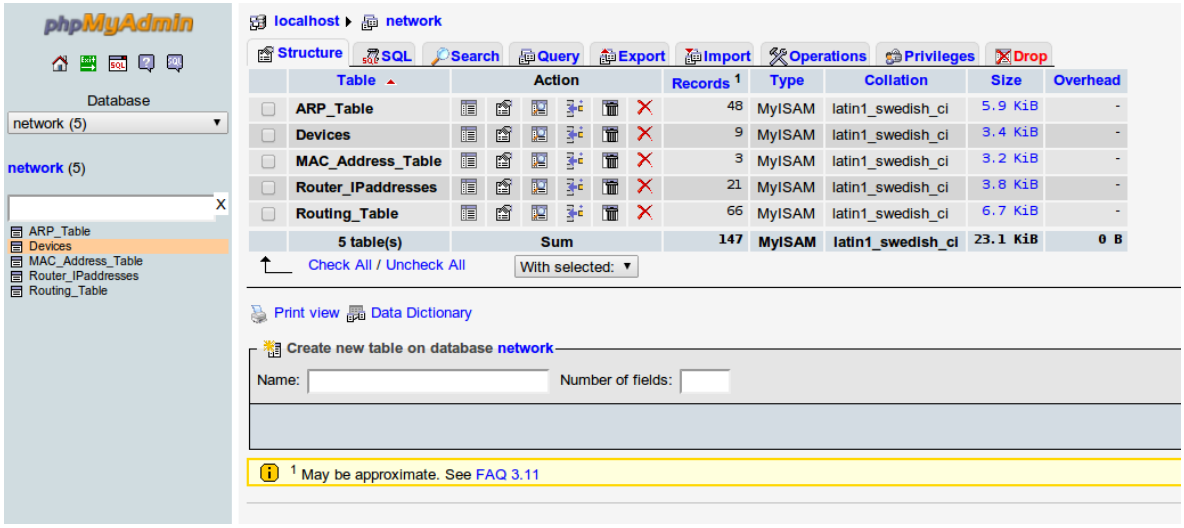


Figure 7.14: Database Tables

Let us analyze the content of each table. First, the Devices Table represented in Fig. 7.15 contains information from all network equipment present in the network that was simulated. Here, it is recorded the device model and manufacturer, device type, hostname and also the device IP address that was used to remotely access its MIB information.

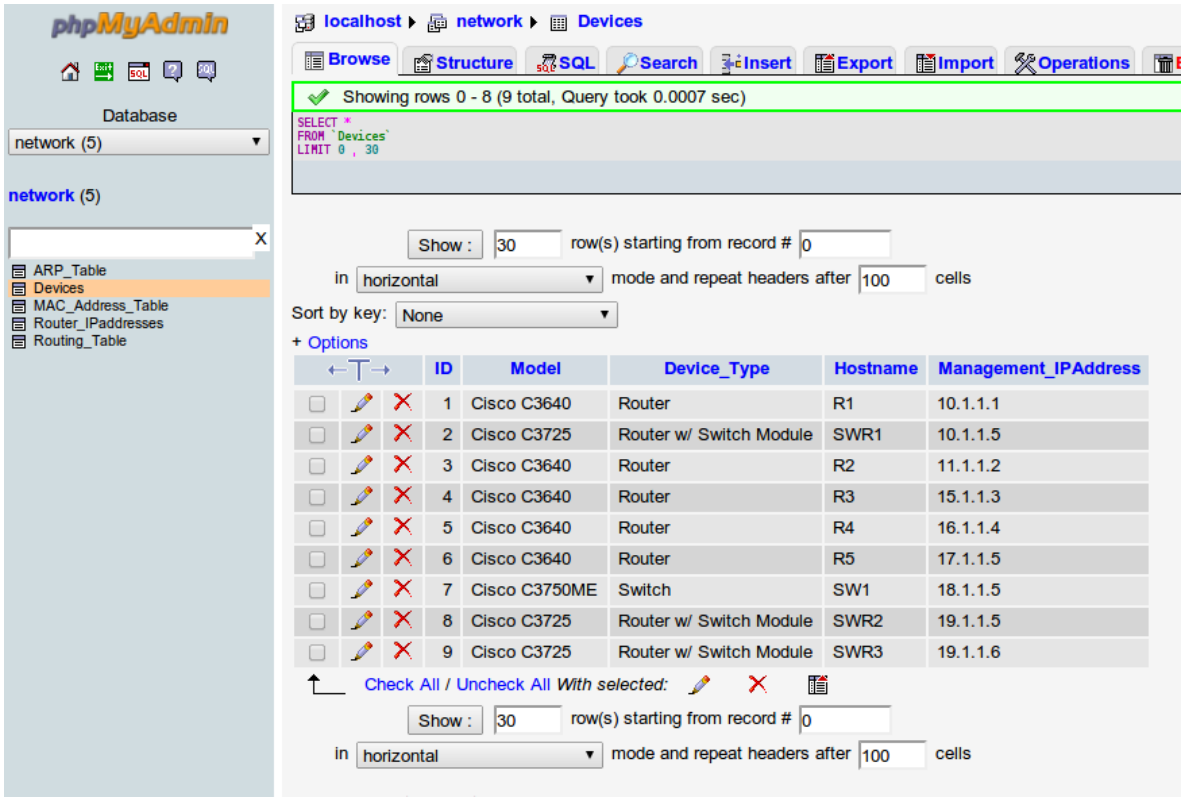
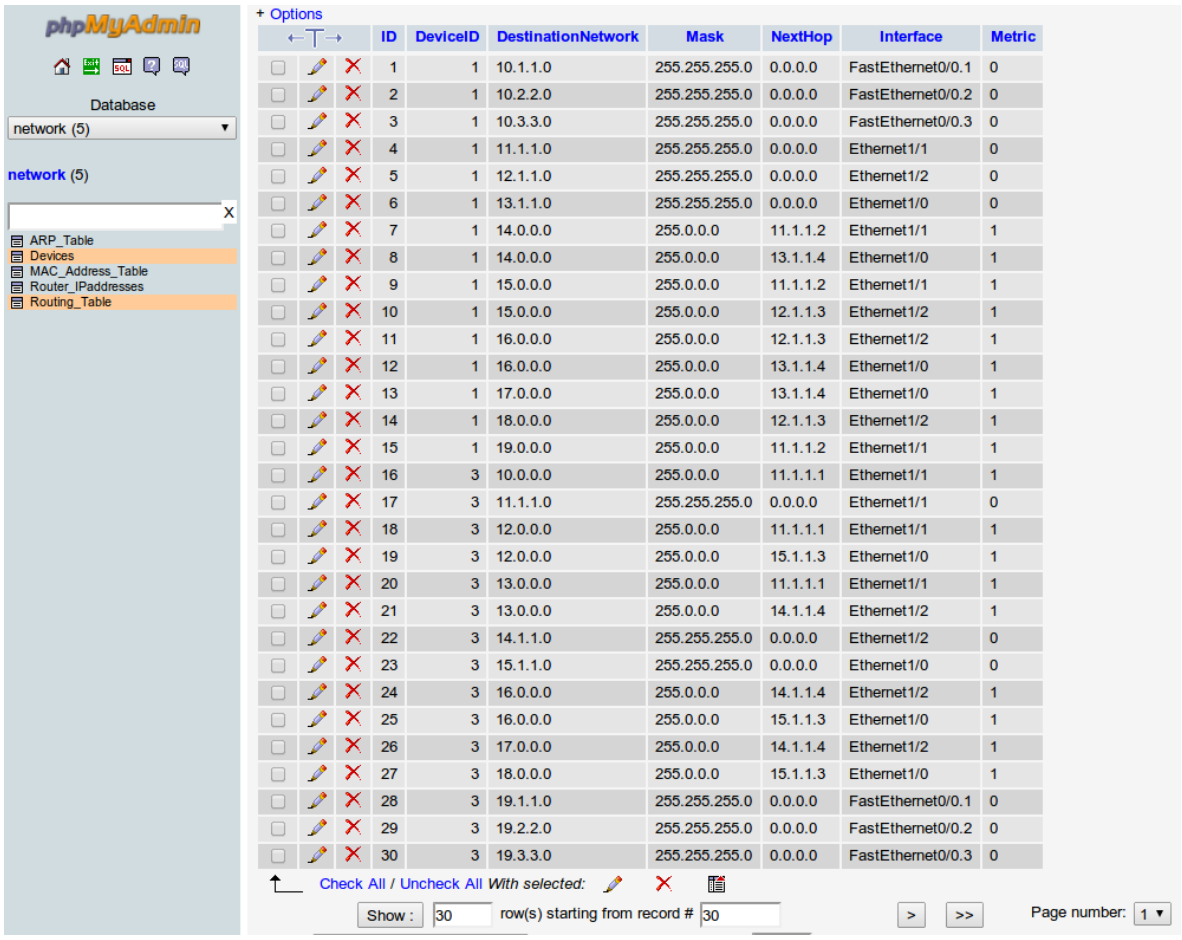


Figure 7.15: Devices Table

Then, the Routing Table is presented in Figures 7.16 and 7.17. Without entering in detail, this table joins all routing information that was displayed for each router, with the detail that the *DeviceID* column represents the device from the Devices Table that each line is referring to.



ID	DeviceID	DestinationNetwork	Mask	NextHop	Interface	Metric
1	1	10.1.1.0	255.255.255.0	0.0.0.0	FastEthernet0/0.1	0
2	1	10.2.2.0	255.255.255.0	0.0.0.0	FastEthernet0/0.2	0
3	1	10.3.3.0	255.255.255.0	0.0.0.0	FastEthernet0/0.3	0
4	1	11.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/1	0
5	1	12.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/2	0
6	1	13.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/0	0
7	1	14.0.0.0	255.0.0.0	11.1.1.2	Ethernet1/1	1
8	1	14.0.0.0	255.0.0.0	13.1.1.4	Ethernet1/0	1
9	1	15.0.0.0	255.0.0.0	11.1.1.2	Ethernet1/1	1
10	1	15.0.0.0	255.0.0.0	12.1.1.3	Ethernet1/2	1
11	1	16.0.0.0	255.0.0.0	12.1.1.3	Ethernet1/2	1
12	1	16.0.0.0	255.0.0.0	13.1.1.4	Ethernet1/0	1
13	1	17.0.0.0	255.0.0.0	13.1.1.4	Ethernet1/0	1
14	1	18.0.0.0	255.0.0.0	12.1.1.3	Ethernet1/2	1
15	1	19.0.0.0	255.0.0.0	11.1.1.2	Ethernet1/1	1
16	3	10.0.0.0	255.0.0.0	11.1.1.1	Ethernet1/1	1
17	3	11.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/1	0
18	3	12.0.0.0	255.0.0.0	11.1.1.1	Ethernet1/1	1
19	3	12.0.0.0	255.0.0.0	15.1.1.3	Ethernet1/0	1
20	3	13.0.0.0	255.0.0.0	11.1.1.1	Ethernet1/1	1
21	3	13.0.0.0	255.0.0.0	14.1.1.4	Ethernet1/2	1
22	3	14.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/2	0
23	3	15.1.1.0	255.255.255.0	0.0.0.0	Ethernet1/0	0
24	3	16.0.0.0	255.0.0.0	14.1.1.4	Ethernet1/2	1
25	3	16.0.0.0	255.0.0.0	15.1.1.3	Ethernet1/0	1
26	3	17.0.0.0	255.0.0.0	14.1.1.4	Ethernet1/2	1
27	3	18.0.0.0	255.0.0.0	15.1.1.3	Ethernet1/0	1
28	3	19.1.1.0	255.255.255.0	0.0.0.0	FastEthernet0/0.1	0
29	3	19.2.2.0	255.255.255.0	0.0.0.0	FastEthernet0/0.2	0
30	3	19.3.3.0	255.255.255.0	0.0.0.0	FastEthernet0/0.3	0

Figure 7.16: Routing Table content - Page 1

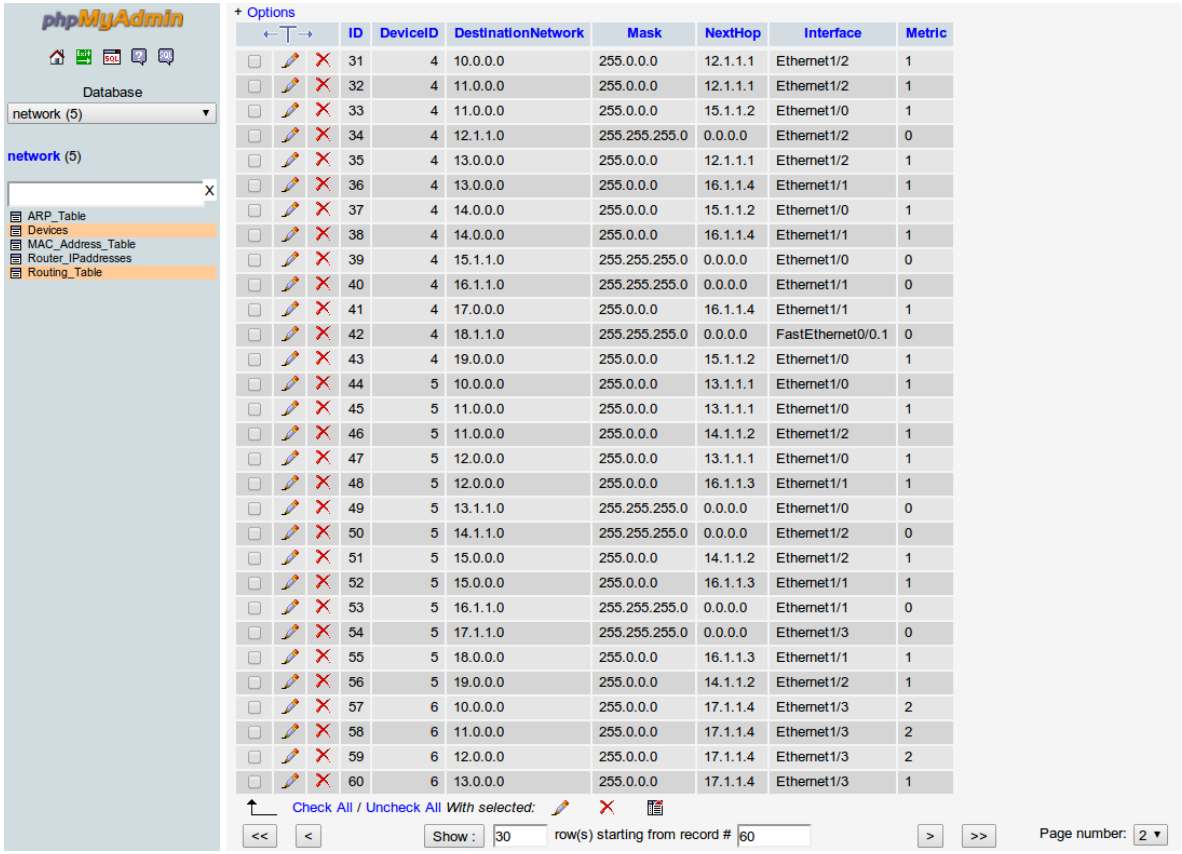


Figure 7.17: Routing Table content - Page 2

Information from the ARP Table of each router was also recorded. This can be verified in Figures 7.18 and 7.19. As in the case of the Routing Tables, these tables also contain all ARP Table information that was previously displayed for each router.

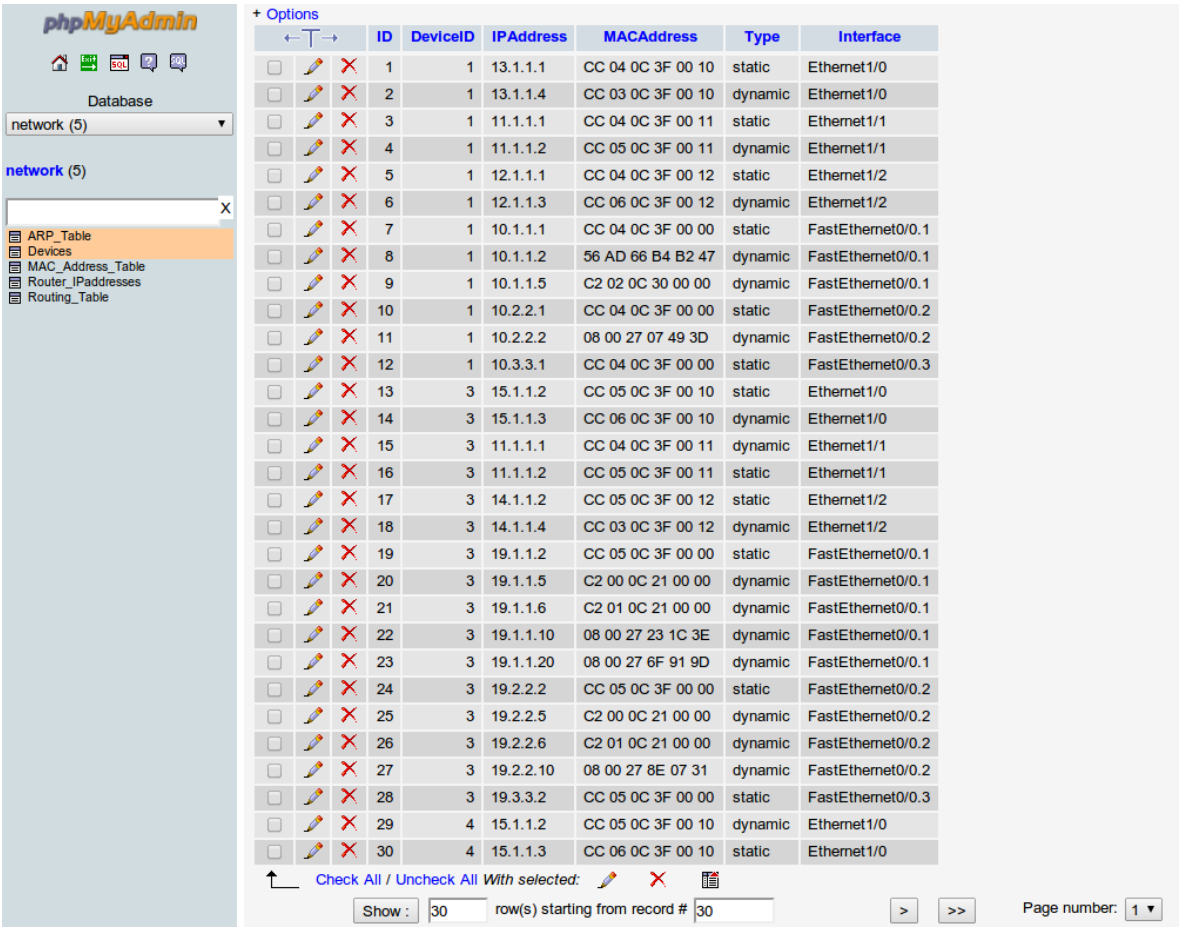


Figure 7.18: ARP Table content - Page 1

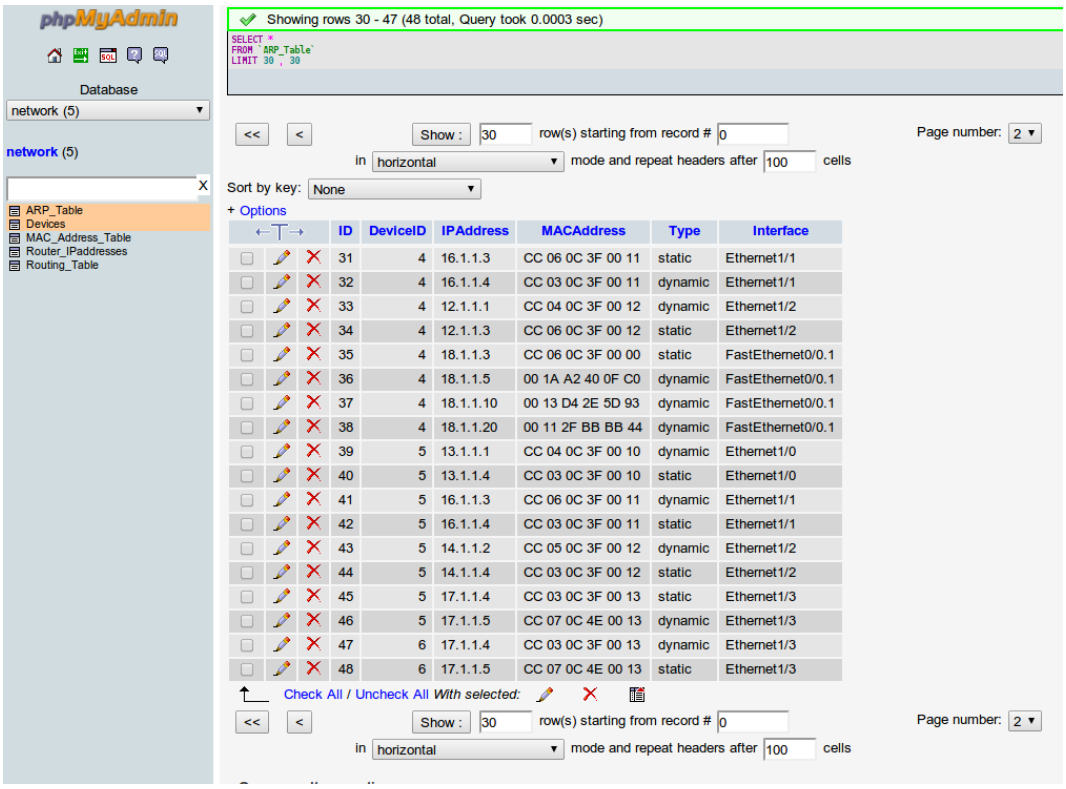


Figure 7.19: ARP Table content - Page 2

The MAC Address Table was also filled with the respective information, as shown in Fig. 7.20. This information was obtained from switch *SW1*, which is the only real switch on the simulated network that could provide this data. Comparing this table with the displayed information from Fig. 7.10, it can be concluded that the information was correctly introduced.

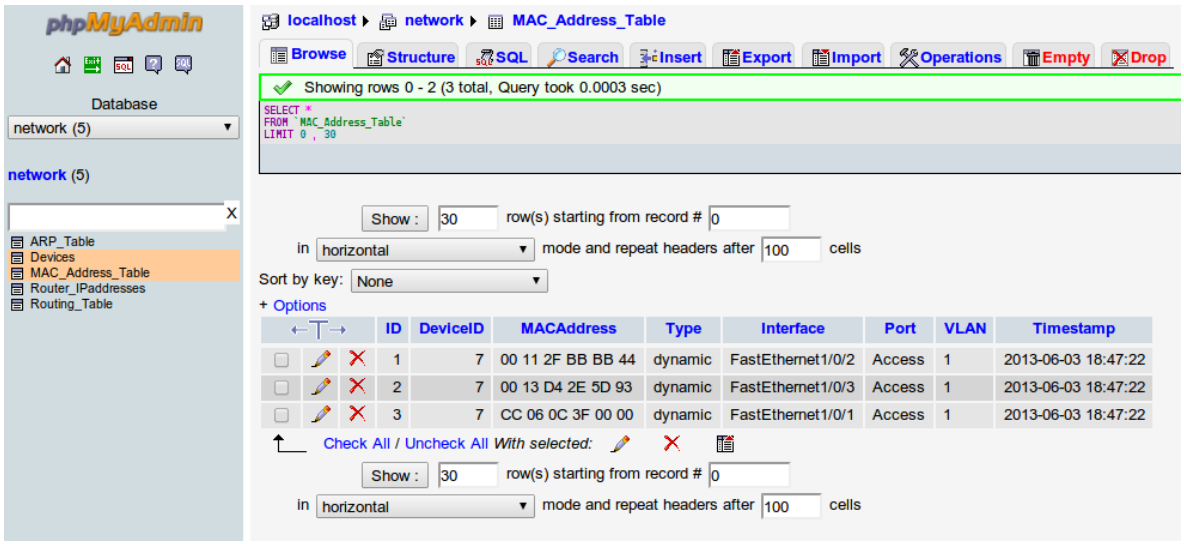
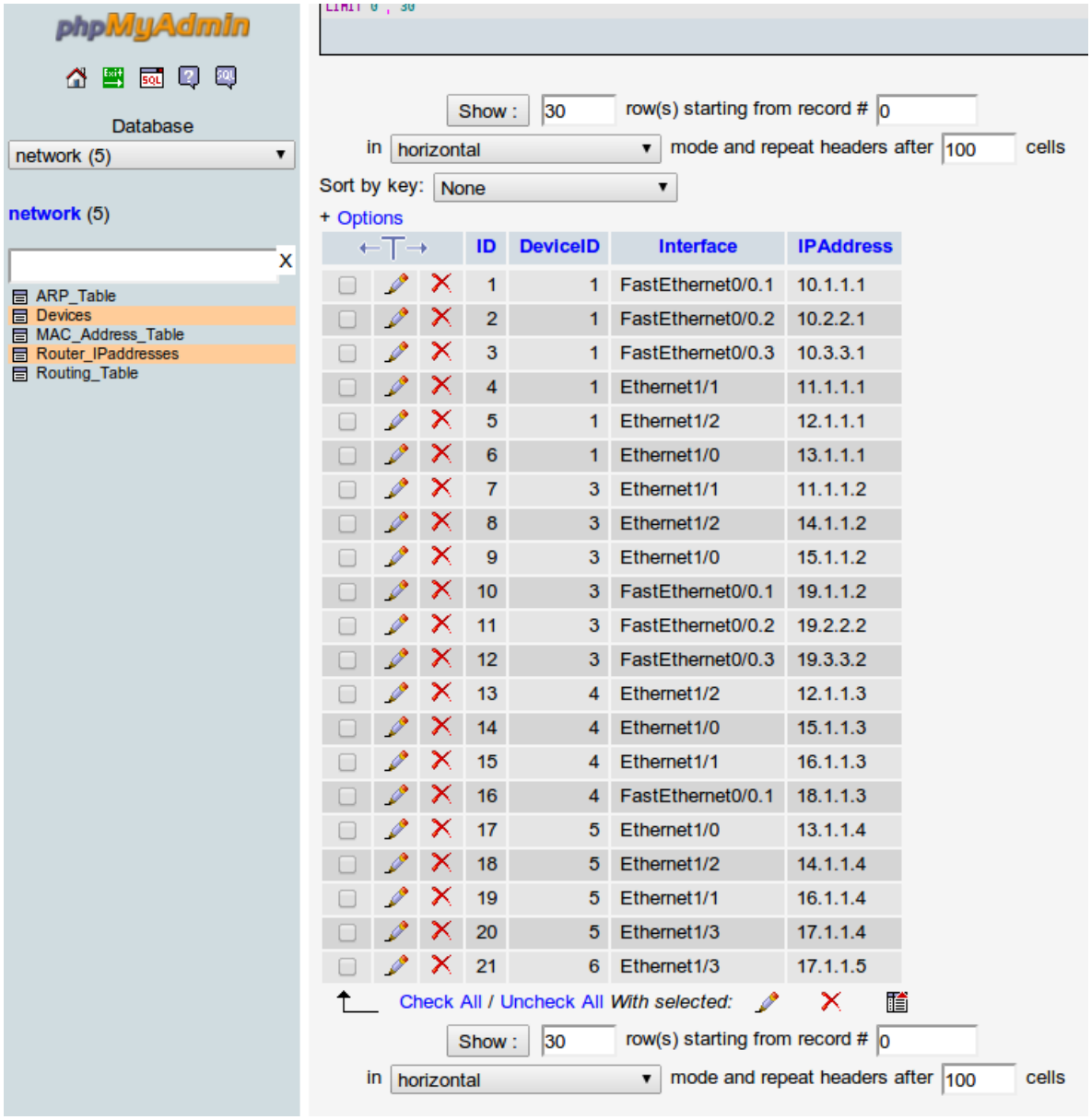


Figure 7.20: MAC Address Table content

Finally, the table containing all IP addresses assigned to each router interface is presented in Fig. 7.21. It's easy to verify the veracity of this data by comparing it with the information that was directly displayed when the algorithm was executed.



ID	DeviceID	Interface	IPAddress
1	1	FastEthernet0/0.1	10.1.1.1
2	1	FastEthernet0/0.2	10.2.2.1
3	1	FastEthernet0/0.3	10.3.3.1
4	1	Ethernet1/1	11.1.1.1
5	1	Ethernet1/2	12.1.1.1
6	1	Ethernet1/0	13.1.1.1
7	3	Ethernet1/1	11.1.1.2
8	3	Ethernet1/2	14.1.1.2
9	3	Ethernet1/0	15.1.1.2
10	3	FastEthernet0/0.1	19.1.1.2
11	3	FastEthernet0/0.2	19.2.2.2
12	3	FastEthernet0/0.3	19.3.3.2
13	4	Ethernet1/2	12.1.1.3
14	4	Ethernet1/0	15.1.1.3
15	4	Ethernet1/1	16.1.1.3
16	4	FastEthernet0/0.1	18.1.1.3
17	5	Ethernet1/0	13.1.1.4
18	5	Ethernet1/2	14.1.1.4
19	5	Ethernet1/1	16.1.1.4
20	5	Ethernet1/3	17.1.1.4
21	6	Ethernet1/3	17.1.1.5

Figure 7.21: IP Addresses Table content

All the results obtained from the execution of the network discovery algorithm was presented. The information was displayed immediately in a readable way for the user and it was also recorded in a database system for a posterior use. To finish the experimental tests related with this algorithm, the case where a stopping network was inserted was tested. Thus, when the information showed in 7.3 was introduced, with network *17.1.1.0* as the stopping network, it was verified that every network device was discovered with the exception of router *R5*. Fig. 7.22 proves this statement by showing the information recorded in the Devices Table

after the algorithm have been executed. It can be observed that now router *R5* is not present in the table. So, in summary, we can consider that all objectives defined for this algorithm were achieved and it was correctly implemented.

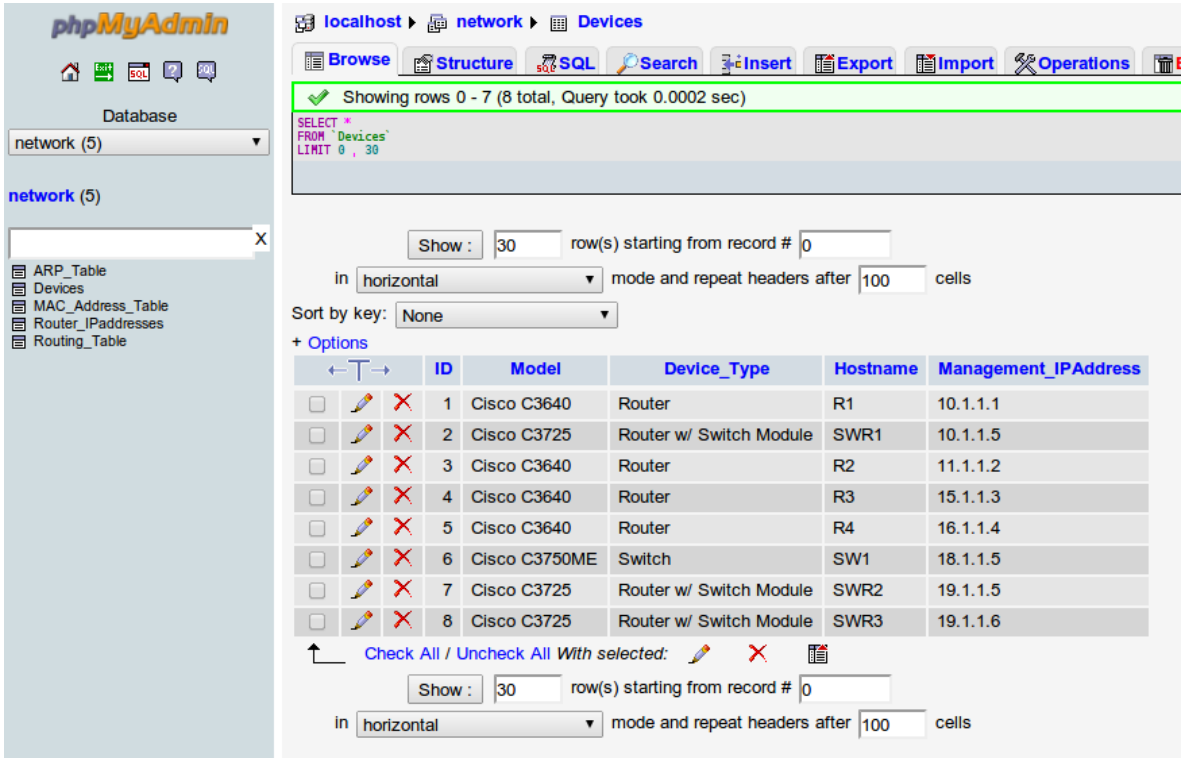


Figure 7.22: Devices Table without R5

The network discovery algorithm allowed not only to consult information from all network devices but also to support the MAC spoofing and IP spoofing detection algorithms so they get knowledge of the presence of these devices to perform the detection task correctly. To test the developed algorithm for detection of MAC spoofing attacks, the corresponding algorithm was executed in an infinite loop. The host used to perform the attack was *PC2* and *PC3* was defined as the victim. Both computers were configured with different IP addresses according to Table 7.1. Then, the MAC address of *PC2* was changed to match the one in use by *PC3*. To change the MAC address of this host, the following commands were executed:

$ sudo ifconfig eth0 down
$ sudo ifconfig eth0 hw ether 00:11:2F:BB:BB:44
$ sudo ifconfig eth0 18.1.1.10 netmask 255.255.255.0 up
$ sudo route add default gw 18.1.1.3 eth0

Basically, these commands shut down the *PC2* interface connected to switch *SW1* and a new MAC address is assigned. Then, this interface is turned on and it is configured with the same IP address and default gateway as previously. This host is now ready to impersonate *PC3* as soon as it starts sending packets to the network.

To simulate a MAC spoofing attack both end hosts will continuously execute *ping* commands to the local computer. When *PC3* starts sending packets, its MAC address is registered

by the algorithm along with other device information according to Fig. 5.3. Then, when *PC2* (the intruder) accesses the network, consecutive changes on the origin of the MAC address are detected. This was the procedure that was taken to simulate MAC spoofing attacks, while the developed detection algorithm was running in background. As shown in Fig. 7.23, it was verified that the attack was actually blocked. The switch interface where the attacker's host (*PC2*) was connected was shut down and its connection to the network got lost. On the other hand, the legitimate host (*PC3*) kept accessing the network without his performance being affected and never losing connection. To confirm the efficiency of this algorithm, 20 attack simulations were performed, which results can be observed in Table 7.2. It was verified that the attacks were detected in 18 of the 20 simulations and once the attacks were detected they were always blocked. The time since the intrusion starts until the intruder's access is blocked was quite variable, with a mean value that falls, with 95% confidence, in the interval [9.368;12.429].

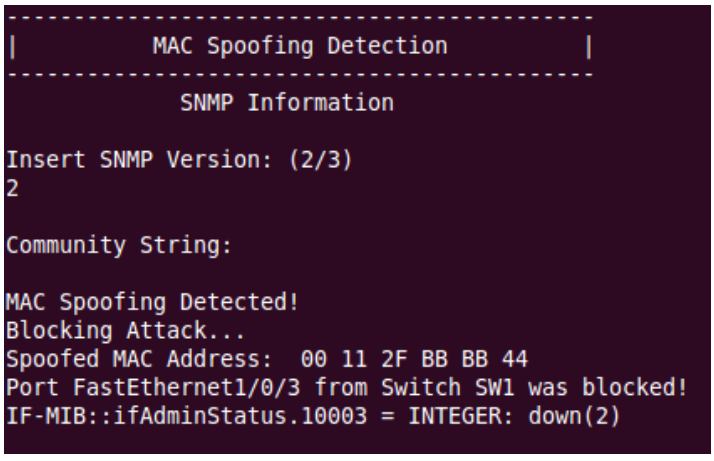


Figure 7.23: MAC spoofing attack detected and blocked

Table 7.2: MAC spoofing attacks simulations

Simulation	Detected	Blocked	Blocking Time (s)
1	✓	✓	12.181
2	✓	✓	7.271
3	✓	✓	8.544
4	×	×	-
5	✓	✓	9.732
6	✓	✓	18.548
7	✓	✓	15.572
8	✓	✓	10.348
9	×	×	-
10	✓	✓	12.835
11	✓	✓	7.649
12	✓	✓	8.640
13	✓	✓	12.248
14	✓	✓	13.800
15	✓	✓	11.561
16	✓	✓	9.825
17	✓	✓	12.216
18	✓	✓	7.459
19	✓	✓	6.836
20	✓	✓	10.909

To test the situation in which the location of the device on the network is changed (Fig. 5.5), the same hosts (*PC2* and *PC3*) were used with the same previous configuration, i.e, both computers configured with different IP addresses and using the same MAC address (*00:11:2F:BB:BB:44*). Thus, instead of changing the actual location of a computer, the two ends hosts will be used as they were a single one that changed location from the place where *PC2* is to the place of *PC3*. This intends to turn the simulation process more efficient. So, having the algorithm running in background, the first host (*PC2*) starts sending packets to the local machine and, after some time, it stops. The MAC address and its origin were registered by the algorithm. After a time period greater than 30 seconds, the second host (*PC3*) executed a *ping* command. The 30 seconds period has to do with the time defined as the border between a possible attack and a change of location. Since they have the same MAC address, this simulates a change on the location of the first host. As it was expected, the new MAC address information was registered and no attack was detected, as shown in Fig. 7.24. This procedure was tested 10 times and in all of them the algorithm didn't interpret it as an attack and the new device location is registered. Thus, it was proved that this method can distinguish between an attack situation when a computer is impersonating another host to access the network and when one device changes its physical location on the network.

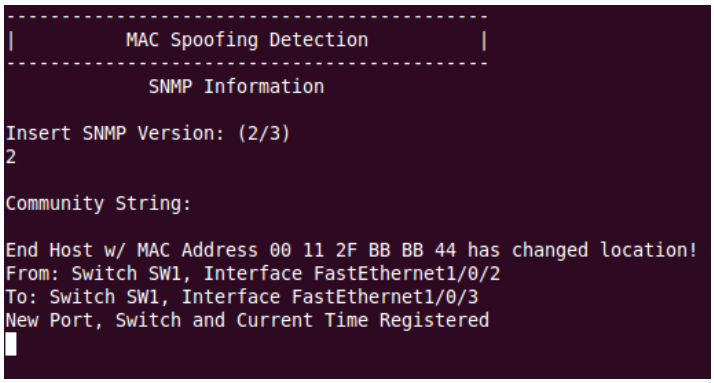


Figure 7.24: End host changes its location in network

Finally, to test the defence mechanism against IP spoofing attacks, the developed algorithm was also executed in an infinite loop. The previous two hosts (*PC2* and *PC3*) were used again, with *PC3* representing the victim's host and *PC2* the intruder one. To simulate the attack, the factory assigned MAC addresses of the two hosts were used, as represented in Table 7.1. In terms of IP addresses, *PC3* used the initially configured IP address (*18.1.1.20*) and *PC2* changed its IP address to match the IP address of the victim's host. Thus, *PC2* was also configured with the IP address *18.1.1.20*.

To perform the simulation tests, we used the same principle as the MAC spoofing simulations. *PC3*, the legitimate client, continually executes *ping* commands to the local machine and *PC2* proceeds the same way. When the first computer starts sending packets, its IP address, MAC address and current time are registered. Then, when the attacker's host accesses the network after a period of time shorter than the 30 minutes that were defined, the algorithm should detect the attack. This was the procedure that was used to simulate IP spoofing attacks with the IP spoofing detection algorithm running in background. As Fig. 7.25 proves, the IP spoofing attack was actually detected. Thus, the switch interface where the intruder was connected was blocked. The performance of *PC3*, the legitimate client, was affected for a few seconds in which a connection outage was verified. However, after this short time period the connection was restored and the user could access the network again. To test the real efficiency of the algorithm, 20 attack simulations were performed. The simulation results can be analyzed in Table 7.3 and it was observed that 20 out of the 20 attacks were detected and all of them were also blocked. In terms of blocking time, it was quite regular, or at least more regular than in the MAC spoofing detection case, with a 95% confidence interval for the mean time equal to [8.426;9.057].

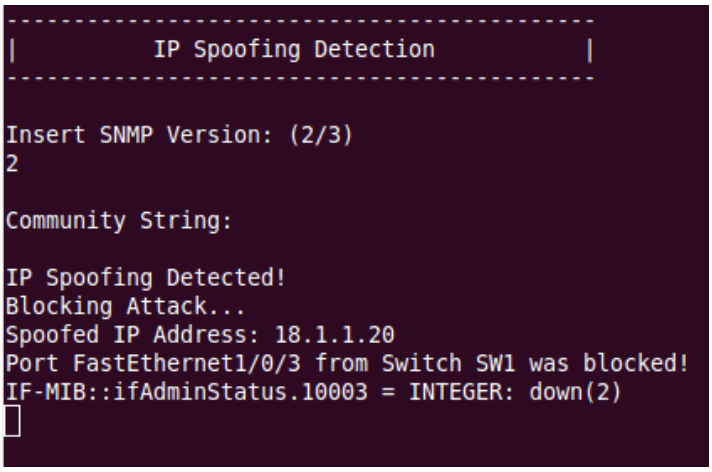


Figure 7.25: IP spoofing attack detected and blocked

Table 7.3: IP spoofing attacks simulations

Simulation	Detected	Blocked	Blocking Time (s)
1	✓	✓	8.079
2	✓	✓	8.352
3	✓	✓	8.469
4	✓	✓	9.042
5	✓	✓	9.040
6	✓	✓	10.848
7	✓	✓	8.612
8	✓	✓	9.292
9	✓	✓	9.076
10	✓	✓	7.536
11	✓	✓	9.071
12	✓	✓	7.863
13	✓	✓	8.795
14	✓	✓	8.920
15	✓	✓	8.613
16	✓	✓	8.424
17	✓	✓	8.560
18	✓	✓	9.264
19	✓	✓	8.452
20	✓	✓	8.517

To test the case in which a second machine is assigned with the same IP address but not with a malicious purpose, a similar procedure was performed with the exception that the first host (*PC3*) starts sending *ping* commands but stops after a while. *PC2* waits a time period of at least 30 minutes after the first one has stopped and then it also starts executing *ping* commands. This will simulate the situation in which a computer is disconnected to

the network while another one gets connected and it is configured with the IP address of the first host. We performed 10 simulations following the previous procedures and in all of them the message presented in Fig. 7.26 was displayed, which means that the new computer information was registered and the algorithm was able to distinguish this situation from a real attack. Thus, the new machine (*PC2*) accessed the network without any problems, while the first one simply got disconnected with no consequences.

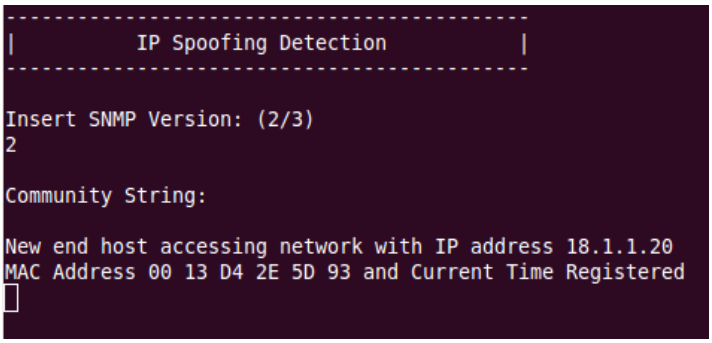


Figure 7.26: New end host in network using an IP that was in use

These experimental tests proved the efficiency of both methodologies for the detection and blocking of MAC spoofing and IP spoofing attacks, even though the first one wasn't completely accurate. The algorithms were also able to distinguish between the situation of a real network attack and when changes on the network were verified. Furthermore, these methods can be easily deployed by simply running the algorithms and they should work in any network since all devices are correctly configured.

Chapter 8

Conclusions and Future Work

After all the network monitoring methodologies have been presented and the corresponding experimental tests have been performed, it is now time to make an overall evaluation about this work and to take the necessary conclusions. First of all, referring to the network discovery algorithm, we had to verify if it could discover any Layer 2 and Layer 3 device on the network that supports SNMP. It was verified, through network simulations, that it actual happens in practice and any router, switch or router using EtherSwitch card is discovered. To have a complete set of experimental tests, the algorithm should have been executed with access points connected to the network too, but unfortunately it wasn't possible. However, theoretically, the algorithm should be able to discover this type of devices without any problem. In terms of information retrieved from each device MIB, after running the algorithm it was possible to observe routing tables and ARP tables from each router present in the network, as well as tables with the interfaces names of each router and the corresponding IP addresses. In the switch case, its MAC address table was observed, as it was supposed. Consulting these tables was one of the defined objectives for this network discovery algorithm and it was correctly implemented. The other objective of this moniroting tool was to record this information in a database for posterior use. As it was seen in the previous chapter, this goal was also achieved, which means that all the objectives set for this monitoring algorithm were successfully achieved.

Then, the MAC spoofing detection methodology was intended to detect any attack performed from a host connected to a switch and block the access of this host to the network by shutting down the switch interface. After running the network simulation, several MAC spoofing attacks were performed and it was observed that this algorithm was able to detect and block 90% of them. It is not working perfectly and certainly not ready to be deployed in a real network, but in any case it is already a good percentage. Other negative aspect was the time that this method took since the attack started until it was detected. Even though the algorithm detects most of the MAC spoofing attacks, the time that it takes is very irregular, which means that sometimes the detection is achieved in a few seconds and in others it takes some more seconds. This is due to the fact that the detection algorithm is based on the information contained in the MAC address table of the switch, which is not regularly updated. But once the attack is detected, the access of the intruder's host was blocked almost instantly. On the other hand, a positive point is that the legitimate client is completely indifferent about what happens in background and it is not affected by the attacks. It was also tested the scenario in which an end host changes its location in the network, whose situation

could lead the algorithm to interpret it as an attack. In all simulations, the algorithm was able to distinguish between this scenario and a real attack scenario. So, in general, it can be considered that the MAC spoofing attack detection algorithm was correctly implemented with a relatively fast and efficient response to the attacks, but with a performance that can still be improved.

Finally, for the IP spoofing attack detection methodology the objectives were similar to the MAC spoofing detection, i.e, detect if an IP address is being spoofed, discover the source of the attack and block the access of the intruder's host to the network. Several IP spoofing attacks against the network were performed and all of them were detected and blocked by the developed algorithm, which means that the primary objective of this algorithm was successfully achieved. The only negative point is the fact that the legitimate host loses connection for a few seconds while the whole process of the intrusion and consequent attack block takes place. Besides this detail, it was obtained a more regular and shorter blocking time when compared to the MAC spoofing blocking, which means that this method is more efficient and accurate. As an overall evaluation of the algorithm performance, it can be considered that this methodology is able to detect any IP spoofing attack and immediately block it. As it was said in the algorithm description, in this method the attacker can be accessing the network from any Layer 2 device. Unfortunately, the algorithm was only tested for attacks performed from a host connected to a switch and it wasn't possible to test its efficiency with APs. It was also tested the scenario in which an end host is disconnected to the network and another host is legitimately configured with the same IP address. The algorithm was able to distinguish this situation and didn't interpret it as an attack. So, as in the case of the MAC spoofing attack detection, here the objective was also achieved and the developed methodology was correctly developed and implemented.

Referring to possible enhancements to the developed work, all the developed algorithms can be somehow improved. In first place, although all implemented scripts are mostly performing their tasks correctly, it's still possible to improve their efficiency in order to perform a faster discovery of the network and also to detect spoofing attacks more rapidly. This is especially important on the second case because the faster the network attacks are detected and blocked, less information the intruder can obtain and less damage is induced on the network. It is also important to have an algorithm with the highest possible accuracy. So, the first improvement would be to rearrange the developed code to obtain faster results and, in the case of the MAC spoofing detection algorithm, to become even more accurate. An interesting functionality that could be added to the project is the creation of a graphical interface to interact with the developed monitoring methodologies. This would allow the user to easily execute the algorithms, introduce the information that is requested and finally to observe the results. The graphical interface would be especially important in the case of the network discovery algorithm because, since the retrieved information is recorded in a database, it would be interesting to have a graphical interface to consult the results and manipulate the information more efficiently instead of observing it directly from the database tables. This would probably be the second feature to add to the project in the future.

The diversity of information contained on each device MIB allows the creation of different monitoring tools. This project developed a methodology for network discovery and two other for detection of spoofing attacks. In the future, some other methodologies to perform different monitoring tasks could be implemented. The great advantage of using SNMP is that, by executing simple commands from a computer, it is possible to consult and manage a wide range of information. For example, information related to data traffic on each device interface

is also available from the MIB. This information could be used to monitor and limit the traffic on certain network connections. In this case, when a limit is reached, the data packets should be redirected through other paths. This is just an example of other monitoring features that could be developed using the SNMP protocol.

In conclusion, the first improvements that could be done should be applied on the developed algorithms. They are all working well but improvements on their efficiency and quickness would be necessary. Besides, the graphical interface would be an important feature to provide an easier interaction to the user. Finally, given the great flexibility of the SNMP protocol, other network monitoring methodologies could be developed for a better control over the network.

Bibliography

[1] N. M. Software, *SNMP Tutorial Part 2: Rounding Out the Basics - OIDs and MIBs*. http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics (Accessed: 22 May 2013).

[2] R. Khare, "Telnet: the mother of all (application) protocols," *Internet Computing, IEEE*, vol. 2, no. 3, pp. 88–91, 1998.

[3] G. Sanjing and H. Lihui, "Research of the telnet remote login," in *In Proceedings of the Third International Symposium on Electronic Commerce and Security Workshops (ISECS 2010)*, pp. 219–221, 2010.

[4] C. M. Kozierok, *The TCP/IP Guide - Telnet Connection and Client/Server Operation*, September 2005. http://www.tcpipguide.com/free/t_TelnetConnectionsandClientServerOperation.htm (Accessed: 2 April 2013).

[5] T. Ylonen, "Ssh - secure login connections over the internet," in *In Proceedings of the 6th USENIX Security Symposium*, pp. 37–42, 1996.

[6] T. Ylonen, *Announcement: Ssh (Secure Shell) Remote Login Program*, July 1995. Original announcement of Ssh.

[7] A. Clemm, *Network Management Fundamentals*, pp. 249–261. Cisco Press, 2007.

[8] C. M. Kozierok, *TCP/IP Simple Network Management (SNMP) Protocol*, September 2005. http://www.tcpipguide.com/free/t_TCPIPSimpleNetworkManagementProtocolSNMPProtocol.htm (Accessed: 2 April 2013).

[9] E. Bibbs and B. Matt, *A Comparison of SNMP v1, v2 and v3*. The Infosec Writers Text Library, April 2006. http://www.infosecwriters.com/text_resources/pdf/SNMP_BMatt.pdf (Accessed: 2 April 2013).

[10] Cisco, *Cisco LAN Management Solution 2.6 Deployment Guide*, 2008. http://www.cisco.com/en/US/prod/collateral/netmgtsw/ps6504/ps6528/ps2425/prod_white_paper0900aecd805441cd.pdf (Accessed: 2 April 2013).

[11] K. S. Nash, *Network Monitoring Definition and Solutions*, June 2009. http://www.cio.com/article/133700/Network_Monitoring_Definition_and_Solutions (Accessed: 15 May 2013).

[12] A. Clemm, *Network Management Fundamentals*, pp. 131–161. Cisco Press, 2007.

[13] L. McKeag, *What can CiscoWorks do for you?*, May 2004. http://howto.techworld.com/networking/563/what-can-ciscoworks-do-for-you/ (Accessed 15 May 2013).

[14] Netcraftsmen, *CiscoWorks LMS 4.0: Improved with More Integration and a New User Interface*, 2013. http://www.netcraftsmen.net/resources/technical-articles/849-ciscoworks-lms-40-improved-with-more-integration-and-a-new-user-interface.html (Accessed: 15 May 2013).

[15] Cisco, *CiscoWorks LAN Management Solution 3.2 and earlier*, 2013. http://www.cisco.com/en/US/products/sw/cscowork/ps2425/index.html (Accessed: 15 May 2013).

[16] Cisco, *CiscoWorks Resource Manager Essentials*, 2013. http://www.cisco.com/en/US/products/sw/cscowork/ps2073/index.html (Accessed: 15 May 2013).

[17] SolarWinds, *About SolarWinds*, 2013. http://www.solarwinds.com/ (Accessed: 17 May 2013).

[18] SolarWinds, *SolarWinds Network Performance Monitor - Powerful Network Fault & Availability Management*, 2013. http://www.solarwinds.com/pdfs/SolarWinds_OrionNPM_Datasheet.pdf (Accessed: 17 May 2013).

[19] SolarWinds, *Orion Network Configuration Manager - Network Configuration & Change Management*, 2013. http://www.solarwinds.com/pdf/SolarWinds_OrionNCM_Datasheet.pdf (Accessed: 17 May 2013).

[20] Nagios, *About Nagios - Overview*, 2013. http://www.nagios.org/about/ (Accessed: 18 May 2013).

[21] Nagios, *Nagios Core*, 2013. http://www.nagios.org/projects/nagioscore/ (Accessed: 18 May 2013).

[22] Nagios, *About Nagios Core*, 2013. http://nagios.sourceforge.net/docs/3_0/about.html (Accessed: 18 May 2013).

[23] Nagios, *Nagios Plugins*, 2013. http://www.nagios.org/projects/nagiosplugins/ (Accessed: 18 May 2013).

[24] Nagios, *Nagios Features*, 2013. http://www.nagios.org/about/features/ (Accessed: 18 May 2013).

[25] G. Bailey and C. Seider, *And then there were few - How to survive the next wave of consolidation among Network Equipment Providers*, 2007. http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-7870-01-nep.pdf (Accessed: 25 May 2013).

[26] Juniper, *Juniper Networks - Products & Services*, 2013. http://www.juniper.net/us/en/products-services/ (Accessed: 25 May 2013).

[27] mibDepot, *412 SNMP SMIv1 and v2 MIBs (51 SMIv1, 361 SMIv2) for Juniper Networks*, 2013. http://www.mibdepot.com/cgi-bin/vendor_index.cgi?r=juniper (Accessed: 25 May 2013).

[28] Alcatel-Lucent, *Alcatel-Lucent: Enterprise and Industries*, 2013. http://www2.alcatel-lucent.com/enterprise-and-industries/ (Accessed: 25 May 2013).

[29] mibDepot, *215 SNMP SMIv1 and v2 MIBs (127 SMIv1, 88 SMIv2) for Alcatel*, 2013. http://www.mibdepot.com/cgi-bin/vendor_index.cgi?r=alcatel (Accessed: 25 May 2013).

[30] Huawei, *Huawei - Corporate Information*, 2013. http://www.huawei.com/en/about-huawei/corporate-info/index.htm (Accessed: 25 May 2013).

[31] mibDepot, *190 SNMP SMIv1 and v2 MIBs (50 SMIv1, 140 SMIv2) for Huawei*, 2013. http://www.mibdepot.com/cgi-bin/vendor_index.cgi?r=huawei (Accessed: 25 May 2013).

[32] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach, "Web spoofing: An internet con game," *Software World*, vol. 28, no. 2, pp. 6–8, 1997.

[33] H. Archana, V. Gauri, and H. Arvind, "Media access control spoofing techniques and its counter measures," in *International Journal of Scientific & Engineering Research*, vol. 2, June 2012.

[34] F. A. D. Gupta, S. B. G. Tiwari, T. C. Y. Kapoor, and F. D. P. Kumar, "Media access control (mac)," 2009.

[35] A. Pandey and J. R. Saini, "Counter measures to combat misuses of mac address spoofing techniques,"

[36] M. Tanase, "Ip spoofing: an introduction," *Security Focus*, vol. 11, 2003.

[37] V. Velasco, "Introduction to ip spoofing," *Retrieved September*, vol. 9, p. 2003, 2000.

[38] S. Puangpronpitag and A. Suwannasa, "A design of egress nac using an authentication visa checking mechanism to protect against mac address spoofing attacks," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on*, pp. 300–303, IEEE, 2011.

[39] H. Wang, C. Jin, and K. G. Shin, "Defense against spoofed ip traffic using hop-count filtering," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 1, pp. 40–53, 2007.

[40] G. Yao, J. Bi, and P. Xiao, "Vase: Filtering ip spoofing traffic with agility," *Computer Networks*, 2012.

[41] E. Sasu and O. Prostean, "Network simulation for mac spoofing detection, using dtf method," in *Applied Computational Intelligence and Informatics (SACI), 2012 7th IEEE International Symposium on*, pp. 291–296, IEEE, 2012.

[42] J. M. Gonzalez, M. Anwar, and J. B. Joshi, "A trust-based approach against ip-spoofing attacks," in *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pp. 63–70, IEEE, 2011.

[43] Y. Ma, "An effective method for defense against ip spoofing attack," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pp. 1–4, IEEE, 2010.

[44] T. Bradley, *Introduction to Intrusion Detection Systems (IDS)*, 2013. http://netsecurity.about.com/cs/hackertools/a/aa030504.htm (Accessed: 20 May 2013).

[45] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *NIST Special Publication*, vol. 800, no. 2007, p. 94, 2007.

[46] Snort, *Snort - What is Snort?*, 2013. http://www.snort.org/ (Accessed: 21 May 2013).

[47] M. Roesch *et al.*, "Snort-lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, pp. 229–238, Seattle, Washington, 1999.

[48] R. Ponnaganti, "Comparative study of three ids systems (nfr, emerald, snort),"

[49] W. Lee, C. T. Park, and S. J. Stolfo, "Automated intrusion detection using nfr: methods and experiences," in *USENIX Intrusion Detection Workshop*, 1999.

[50] C. Childers, L. Bangert, and M. O'Connor, "Tracking web usage with network flight recorder.," in *WebNet* (H. A. Maurer and R. G. Olson, eds.), AACE, 1998.

[51] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling response to anomalous live disturbances," in *Proceedings of the 20th national information systems security conference*, pp. 353–365, 1997.

[52] SRI-International, *EMERALD - Project Description*, 2000. http://www.csl.sri.com/projects/emerald/project.html (Accessed: 22 May 2013).

[53] Cisco, *SNMP Object Navigator*, 2013. http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en (Accessed: 20 February 2013).

[54] Y. Qiuxiang, "Algorithm research of topology discovery on snmp," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 12, pp. V12–496, IEEE, 2010.

[55] K. Qin and C. Li, "Network topologic discovery based on snmp," in *Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on*, pp. 1–3, IEEE, 2010.

[56] Cisco, *Using SNMP to Find a Port Number from a MAC Address on a Catalyst Switch*, 2013. http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a00801c9199.shtml (Accessed: 13 April 2013).

[57] D. Spiewak, *Defining High, Mid and Low-Level Languages*, February 2008. http://www.codecommit.com/blog/java/defining-high-mid-and-low-level-languages (Accessed: 2 April 2013).

[58] F. S. Foundation, *Bash Reference Manual*, August 2012. http://www.gnu.org/software/bash/manual/bash.html (Accessed: 4 April 2013).

[59] C. Ramey, *BASH The Bourne-Again Shell*, March 2013. http://tiswww.case.edu/php/chet/bash/bash-intro.html (Accessed: 4 April 2013).

[60] G. Harrison, *10 things you should know about NoSQL databases*, August 2010. http://www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772 (Accessed: 7 April 2013).

[61] J. Cogswell, *SQL vs. NoSQL: Which Is Better?*, July 2012. http://slashdot.org/topic/bi/sql-vs-nosql-which-is-better/ (Accessed: 7 April 2013).

[62] Oracle, *Top Reasons to Use MySQL*, 2013. http://www.mysql.com/why-mysql/topreasons.html (Accessed: 8 April 2013).

[63] Oracle, *MySQL 5.1 Reference Manual: History of MySQL*, 2013. http://dev.mysql.com/doc/refman/5.1/en/history.html (Accessed: 7 April 2013).

[64] phpMyAdmin, *phpMyAdmin: Features*, 2013. http://www.phpmyadmin.net/home_page/index.php (Accessed: 8 April 2013).

[65] Cisco, *SNMP Object Navigator - sysObjectID*. http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&translate=Translate&objectInput=1.3.6.1.2.1.1.2 (Accessed: 15 April 2013).

[66] A. Pandey and J. R. Saini, "Counter measures to combat misuses of mac address spoofing techniques,"

[67] S. S. Rana and T. Bansod, "Ip spoofing attack detection using route based information," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 4, pp. pp–285, 2012.

[68] C. N. Academy, *Cisco Packet Tracer: At-A-Glance*. Cisco, 2010. http://www.cisco.com/web/learning/netacad/course_catalog/docs/Cisco_PacketTracer_AAG.pdf (Accessed: 5 April 2013).

[69] C. N. Academy, *Cisco Packet Tracer Data Sheet*. Cisco, 2010. http://www.cisco.com/web/learning/netacad/course_catalog/docs/Cisco_PacketTracer_DS.pdf (Accessed: 5 April 2013).

[70] Boson, *Boson - About us*, 2013. http://www.boson.com/about-us (Accessed: 5 April 2013).

[71] J. Harry, *Using the GNS3 Network Simulator*, March 2010. http://www.trainsignal.com/blog/using-gns3-network-simulator (Accessed: 5 April 2013).

[72] GNS3, *Introduction to GNS3*, 2013. http://www.gns3.net/documentation/gns3/introduction-to-gns3/ (Accessed: 5 April 2013).

[73] GNS3, *Memory and CPU Usage*, 2013. http://www.gns3.net/documentation/gns3/memory-and-cpu-usage/ (Accessed: 5 April 2013).

[74] Cisco, *Configuring EtherChannel and 802.1Q Trunking Between Catalyst L2 Fixed Configuration Switches and a Router (InterVLAN Routing)*. http://www.cisco.com/en/US/products/hw/switches/ps628/products_configuration_example09186a00800ef797.shtml (Accessed: 10 April 2013).

[75] Cisco, *Configuring Routing Information Protocol*. http://www.cisco.com/en/US/docs/ios/12_2/ip/configuration/guide/1cfrip.html (Accessed: 10 April 2013).

[76] M. Krasnyansky, M. Krasnyansky, and M. Yevmenkin, *The Linux Kernel Archives - TAP documentation*, 2000. https://www.kernel.org/doc/Documentation/networking/tuntap.txt (Accessed: 10 May 2013).

[77] GNS3, *Switching simulation in GNS3*, November 2012. http://www.gns3.net/documentation/gns3/switching-simulation-in-gns3/ (Accessed 20 April 2013).

[78] Boson, *NetSim: The Cisco Network Simulator & Router Simulator*, 2013. http://www.boson.com/netsim-cisco-network-simulator (Accessed: 5 April 2013).