Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

CARLOS MIGUEL DOS SANTOS MIRANDA

IMPLEMENTAÇÃO DE CENARIOS PARA OBTENÇÃO DE CONHECIMENTO ABSOLUTO DE REDE

IMPLEMENTATION OF REALISTIC SCENARIOS FOR GROUND TRUTH PURPOSES

Universidade de Aveiro
2009
Departamento de Electrónica, Telecomunicações e Informática

CARLOS MIGUEL DOS SANTOS MIRANDA

IMPLEMENTAÇÃO DE CENARIOS PARA OBTENÇÃO DE CONHECIMENTO ABSOLUTO DE REDE

IMPLEMENTATION OF REALISTIC SCENARIOS FOR GROUND TRUTH PURPOSES

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor. Paulo Salvador, Professor Auxiliar e do Doutor. António Nogueira, Professor Auxiliar, ambos do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico esta dissertação à minha família

o júri

presidente

Prof. Doutor Aníbal Manuel Oliveira Duarte
Professor Catedrático da Universidade de Aveiro

Prof. Doutor Rui Jorge Morais Tomaz Valadas
Professor Catedrático do Departamento de Engenharia Electrotécnica e de Computadores do Instituto Superior Técnico - Universidade Técnica de Lisboa

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

Prof. Doutor António Manuel Duarte Nogueira
Professor Auxiliar da Universidade de Aveiro

agradecimentos

Aos meus pais, Luís de Jesus dos Santos e Maria de Lurdes Miranda de Pinho, aos meus irmãos, Luís Manuel dos Santos e Daniel Alberto dos Santos e à minha cunhada, Rosangel Pernalete, pelo apoio dado ao longo dos anos que me motivou e não me deixou desistir.

A todos os meus amigos que, de uma forma ou outra, desde sempre me apoiaram, em especial ao Eduardo Rocha e André Azevedo que, com os seus conhecimentos e sugestões, me ajudaram com alguns problemas encontrados durante a realização deste trabalho e me apoiaram nas alturas mais complicadas ao longo deste ano e à Sónia Luís, Inês Carvalho, Luís Alves, Miguel Souto, Pedro Borges e Luís Borges pela sua incondicional amizade e apoio ao longo de todos estes anos. Muito obrigado a todos.

À minha namorada, Ana Gomes, que nestes últimos meses tem sido o meu apoio e uma força inspiradora para poder continuar e que me fez acreditar mais um pouco em mim.

Ao meu orientador, Prof. Doutor Paulo Jorge Salvador Serra Ferreira, e ao meu co-orientador, Prof. Doutor António Manuel Duarte Nogueira, cujos conhecimentos e sugestões foram fundamentais para a realização desta dissertação.

Ao Prof. Doutor Rui Jorge Morais Tomaz Valadas pelo seu conhecimento e ideias sobre o tema proposto.

A todas estas pessoas, um muito obrigado pelo apoio e confiança que depositaram em mim.

Esta dissertação foi elaborada também com o apoio do Instituto de Telecomunicações, que disponibilizou o equipamento necessário para a realização deste trabalho.

palavras-chave

Rede, emulação, simulação, segurança, topologias de rede, serviços, equipamentos de rede, *ground truth*.

resumo

A segurança em redes de telecomunicações é um tópico que desde sempre gerou preocupação em todos os meios (instituições, empresas e outros) que utilizam estas redes. Novas ameaças ou mutações de ameaças já existentes surgem a uma elevada velocidade e os meios disponíveis parecem não ser suficientes para uma detecção positiva das mesmas.

As respostas actuais para combater estas ameaças baseiam-se numa análise em tempo real do tráfego ou num treino prévio que muitas vezes tem que ser supervisionado por um ser humano que, dependendo da sua experiência na área pode estar a criar uma falha de segurança no sistema sem se aperceber do sucedido.

Novas técnicas surgem para uma detecção eficaz de muitos ataques ou anomalias. No entanto, estas técnicas devem ser testadas de modo a validar o seu correcto funcionamento e, nesse sentido, são precisos fluxos de tráfego gerados na rede que possam ser utilizados sem comprometer a confidencialidade dos utilizadores e que obedeçam a critérios pré-estabelecidos.

Com esta dissertação pretende-se constituir um conjunto de dados fiável e o mais abrangente possível de um conjunto de cenários realistas de rede, através da emulação em ambiente controlado de diferentes topologias, diferentes serviços e padrões de tráfego. Um outro objectivo fundamental deste trabalho passa por disponibilizar os dados obtidos à comunidade científica de modo a criar uma base de dados uniforme que permita avaliar o desempenho de novas metodologias de detecção de anomalias que venham a ser propostas.

keywords

Network, emulation, simulation, security, network topologies, network equipment, ground truth.

abstract

Security in telecommunication networks is a topic that has caused a lot of worries to network users (institutions, enterprises and others). New threats or mutations of existing ones appear at a very fast rate and the available solutions seem not to be enough for a positive detection of these threats.
The solutions that are nowadays used to fight these threats require the real-time analysis of the network traffic or have to be previously trained. Most of the times, this training has to be supervised by a human being that, depending on his experience, can create a security breach in the system without knowing it.
New techniques have been proposed in order to more efficiently detect many security attacks or threats. However, these techniques need to be tested in order to validate their correct functioning and, in order to do that, network traffic flows that can be used without compromising the users confidentiality and that obey to a pre-established criteria are needed.
This dissertation intends to establish a set of trustworthy data as extensive as possible from a set of realistic network scenarios. Network emulation techniques will be used in a controlled environment, building different network topologies, with different services and traffic patterns. Another main objective of this work it is to make all this obtained data available to the scientific community in order to create a uniform data base that will allow the performance evaluation of new anomaly detection methodologies that can be proposed in the future.

Table of contents

List of Figures

List of Tables

List of Acronyms and Glossary

- *ARCA* – Arquivo Central de Ficheiros. FTP site in the University of Aveiro.
- *ATM* – Asynchronous Transfer Mode.
- *BIND9* - Berkeley Internet Name Domain version 9.
- *CCNA* – Cisco Certified Network Associate.
- *CCNP* – Cisco Certified Network Professional.
- *CLI* – Command Line Interface.
- *DARPA* – Defense Advanced Research Projects Agency.
- *DHCP* – Dynamic Host Configuration Protocol.
- *Diffserv* – Differentiated Services. End-to-End Quality of Service Model.
- *DNS* – Domain Name System.
- *ESMTP* - Extended Simple Mail Transfer Protocol.
- *FIFO* – First In, First Out.
- *FreeBSD* – Operative system based on the UNIX platform mainly used on servers.
- *FTP* – File Transfer Protocol.
- *FTPS* - File Transport Protocol Secure.
- *GNS* – Graphical Network Simulator.
- *GPRS* – General Packet Radio Service.
- *GTNetS* – Georgia Tech Network Simulator.
- *GUI* – Graphical User Interface.
- *HTTP* – Hypertext Transfer Protocol.
- *HTTPS* - Hypertext Transfer Protocol Secure.
- *IDS* – Intrusion Detection System.
- *IOS* – Internetwork Operating System.
- *IP* – Internet Protocol.
- *Ipfw* – IPFIREWALL. FreeBSD's Firewall.
- *IPX* – Internetwork Packet Exchange.
- *ISI* – Information Science Institute.
- *LAN* – Local Area Network.
- *LSA* - Link State Advertisements.
- *LSU* - Link State Update.
- *M-file* – Script file that uses Matlab commands.
- *MAC* – Media Access Control.
- *MD* - Managed Devices.
- *MIB* – Management Information Base.

- *MIT* – Massachusetts Institute of Technology.
- *NCTU* – National Chiao Tung University.
- *NED* – Network Description Language.
- *NIC* - Network Interface Card.
- *NMS* - Network Management System.
- *OpenWRT* – Linux based firmware program for embedded devices.
- *OSPF* – Open Shortest Path First.
- *OTcl* – MIT Object Tcl. An extension to Tcl/Tk for object oriented programming.
- *PHP* – Hypertext Preprocessor.
- *POP3* – Post Office Protocol Version 3.
- *QoS* – Quality of Service.
- *RED* – Random Early Detection Queue Management.
- *SMTP* – Simple Mail Transfer Protocol.
- *SNMP* – Simple Network Management Protocol.
- *SQL* – Structured Query Language.
- *Tcl/Tk* – Tool Command Language/Toolkit.
- *TCP* – Transmission Control Protocol.
- *TELNET* – Teletype Network.
- *UDP* – User Datagram Protocol.
- *URL* - Uniform Resource Locator.
- *USC* – University of Southern California.
- *VIP/VIPA* - Virtual IP Address.
- *VPCS* – Virtual PC Simulator.
- *WAN* – Wide Area Network.
- *WF^{2}Q++* - *Worst*-case Fair Weighted Fair Queue.
- *Worm* - Self-replicating virus that does not alter files.
- *WWW* - World Wide Web.

1. Introduction

Security in telecommunication networks is a topic that has caused a lot of worries to network users (institutions, enterprises and others). New threats or mutations of existing ones appear at a very fast rate and the available solutions seem not to be enough for a positive detection of these threats.

The solutions that are nowadays used to fight these threats require the real-time analysis of the network traffic or have to be previously trained. Most of the times, this training has to be supervised by a human being that, depending on his experience, can create a security breach in the system without knowing it.

New techniques have been proposed in order to more efficiently detect many security attacks or threats. However, these techniques need to be tested in order to validate their correct functioning and, in order to do that, network traffic flows that can be used without compromising the users confidentiality and that obey to a pre-established criteria are needed.

One possible way to test these techniques could rely on knowing the Ground Truth of the topology, but obtaining the Ground Truth using real network traffic flows has proven to be an almost impossible task. In fact, in order to obtain the Ground Truth we need to know (i) the topology of the network, which can be a difficult task due to the heterogeneity factor of the Internet; (ii) the "before and after" of the network traffic flows that were used and (iii) exactly how many anomalies are present in the traffic flow or if it is an anomaly-free traffic flow.

Another important factor is the confidentiality problem. It is difficult for researchers to find or share traces and results because it is impossible to guarantee that the networks traces do not contain any confidential information from the users so it is preferred not to share the information gathered not only for the protection of the users, but for the researchers themselves.

This dissertation will try to prove that it is possible to create an offline emulation platform that is able to test anomaly detection techniques using traffic flows generated inside the platform and obtaining the Ground Truth corresponding to a given topology.

1.1. Motivation

The motivation behind this work lies in the scientific community need of an offline emulation environment that is capable of generating high quality, real network traffic to analyze the behavior of the different elements inside the network and to analyze the behavior of security attacks.

As previously said, new identification techniques have been proposed in the last years, but in order to test their efficiency, they have to analyze network traffic flows that can often be of poor quality or have confidential information, making almost impossible the task of testing their accuracy or sharing the obtained information and findings.

1.2. Objectives

The main objective of this dissertation is to create a controlled emulation environment or emulation lab, that can support multiple network nodes, moderate sized network topologies, multiple emulated end-hosts and services, and that can also connect to real network equipment (routers, layer 3 switches, servers and others) in order to analyze the behavior and trace the profile of emulated users and security attacks. By doing this, we expect to prove that is possible to obtain the high quality data that is needed to test new anomaly detection techniques without putting at risk any confidential data.

Another strategic objective for this work is to make all the obtained data available to the scientific community in order to create a uniform data base that will allow the performance evaluation of new anomaly detection methodologies that can be proposed in the future.

As stated before, the main objective is to create a controlled environment where the Ground Truth of a given topology can be obtained. The results shown in this dissertation are not intended to be a definite set of trustful network data, but will only serve as an example of the data that can be gathered in this platform.

1.3. Organization of this dissertation

This dissertation is divided into six chapters:

- Chapter 1 - **Introduction.**
- Chapter 2 - **Background**: this chapter presents the definition of Ground Truth, its characteristics and requirements, the importance of Ground Truth in the network security area, the problems of using public data and the difficulties of simulating the Internet. Besides, this chapter also presents the definitions of simulators and emulators, their main characteristics and types and the reason for using an emulator in this dissertation.
- Chapter 3 - **GNS3:** this chapter explains the emulation software GNS3 that will be used in this dissertation, its main characteristics and the virtual devices that are needed to connect an emulated scenario inside GNS3 to real equipment.
- Chapter 4 - **Emulation:** this chapter will present all the necessary configurations that are required in the server and in the PCs where the scenarios are emulated and, also, the scenarios that will be emulated in this dissertation.
- Chapter 5 - **Collected data:** this chapter shows examples of the different data that was gathered to test the correct functioning of the emulation platform. It also shows the profiles of the different services implemented in the emulated network. Besides, this chapter will also show the different log files and syntaxes for all the services that were installed in the server.
- Chapter 6 - **Conclusion and Future Work:** this chapter shows the main conclusions of this work and presents some suggestions for future work.

2. Background

2.1. Introduction

Security has been a hot topic for years. Many efforts have been made over the years to minimize the impact of anomalies that may cause severe damages to all types of internet users, ranging from normal home internet users to institutional network administrators. The monitoring of the network and the usage of Intrusion Detection Systems (IDS) represent the actual solution to the majority of security problems that can be found on the internet, but they have also some weak points:

- Almost all security solutions require real-time training and this training is made by asking questions to the users about specific events. For trained users this is normally not a problem but for beginners this could be the point where everything goes wrong. Even for trained users it can go wrong here if they are distracted and make the mistake of accepting (or not) a request. By doing this, they could be opening a breach in their system that a hacker can later take advantage of.
- The need for monitoring the network makes it very difficult for a specific IDS to classify all the traffic in real-time and analyze every single aspect of the traffic flow without having some kind of "database" of the characteristics it should be looking for.
- The number of "False-Positive" events is very high when training some of the solutions.
- Since they are based on a specific "signature" to classify the anomaly, some IDS are not capable of detecting mutations of the anomaly (and in this area, anomalies are a very fast moving target) because they look only for the "signature" rather than looking for a kind of "behaviour out of the ordinary".

A lot of research has been conducted to develop new methodologies and solutions to automatically detect anomalies, such as worms and DDoS attacks[2, 4, 8]. These algorithms detect variations from a "normal" behaviour instead of searching for the signature of the anomaly, but they need to be tested in order to evaluate their effectiveness. In order to test their precision, one needs to know which anomalies can be detected by these solutions and then use a set of proper anomalies that should be considered by those solutions as "true anomalies".

One way of doing this identification is to use a real trace and manually label some events as being "true-positive anomalies" and see if the detector also classifies them as anomalies. Normally this manual labelling is done by experts but this method brings some problems:

- The impossibility of sharing the traces with the rest of the scientific community due to privacy concerns.
- Even the best experts in the world can miss a true anomaly in a trace.
- The process can't scale to the magnitude that is necessary to identify rare anomalies.
- *The use of a specific trace prevents from performing a sensitivity analysis to know, for example, how big must be an anomaly in order to be detected*.[9]
- There are no guarantees that a "public trace" is really a "clean trace". In other words, if a trace is really free of anomalies, a specific anomaly can be injected in order to compare the differences between them.

But what if we could have these kinds of traces? Clean traces and traces with specific anomalies that could be shared without the concern of putting at risk private data. Traces based on realistic and actual network scenarios with all the active protocols needed to emulate real operations in a network. This is where obtaining the "Ground Truth" in a simulated (in the case of this work, emulated) and controlled environment comes in handy.

2.2. Ground Truth

Ground Truth is a technique that is used in many areas that need to collect data. This is done by performing a series of measurements of various properties in the obtained data set. In terms of network anomaly detection, it requires a list of all the existing anomalies in the data set that is being analysed. This seems to be an easy task to accomplish but in fact, it is not that easy when using real traces. Some of the problems that can be encountered when using real traces are:

- There are no guarantees that the real traces are, in fact, traces without any kind of anomalies. This could be useful in terms of comparison, when one needs to look for behaviours that are out of the ordinary.

- Poor quality data. To identify anomalies, it is necessary to investigate and look at a vast amount of data and when this data has a poor quality (because of data-reduction techniques, for example) a correct identification of the anomalies cannot be guaranteed.

In order to try to solve these problems, some projects [4] have injected anomalies in real traces taken from operational networks but this technique cannot provide truthful measurements because it relies on existing traces and as said before, there are no guarantees that these traces are free of anomalies or, if knowing that they are not clean, on the numbers of anomalies that those traces have.

Another requirement in Ground Truth is the detailed information of the anomaly in terms of:

- Location;
- Magnitude and
- Type.

The location is divided into two categories, spatial and temporal. With the spatial information, an anomaly detector can discover if anomalies that appear to be separated are correlated and the temporal information (time and duration of an anomaly) can be used to calculate the detection delay.

The magnitude of an anomaly is used to test the sensitivity of a detector to a certain type of anomaly.

Different detectors will excel at finding various types of anomalies [9]. Knowing the type of the anomaly is very important and a lot of efforts have been made in previous works to develop detectors that could be able to identify specific anomalies [2, 8]. These types of detectors will have good results identifying the anomalies for what they were developed but they will have poor results with other types of anomalies.

As previously mentioned, these problems can be avoided by using emulators or simulators to obtain all the required data. By defining the topologies of the scenarios, the protocols to be used and the anomalies that we want to test, we can obtain a vast amount of data. We can see the "before and after" the injection of an specific anomaly in order to see the behaviour over the network and we can have a great accuracy in the obtained data without compromising private data, which is one of the main problems of sharing traces with the rest of the community. But using these tools also brings out another difficulty: how can we simulate (or emulate) something as vast and complex as the Internet?

2.3. Simulating the Internet

The Internet is a vast and complex world. As it can be seen in the figures 1.1 and 1.2, the latest statistics provided by the "Internet World Stats" [12] show that the actual number of Internet users is more than 1,6 billion and the majority of the users are in the Asian continent. This number of Internet users is nearly 25% of the global population.

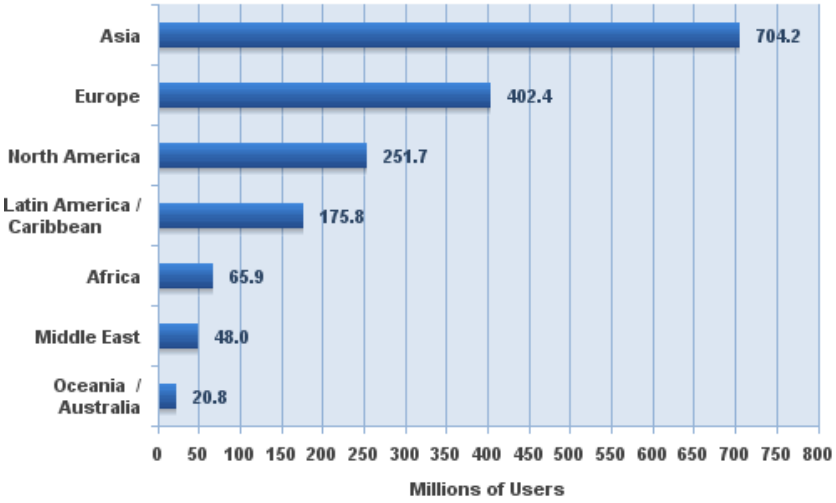


Fig. 2.1 - Internet users around the world

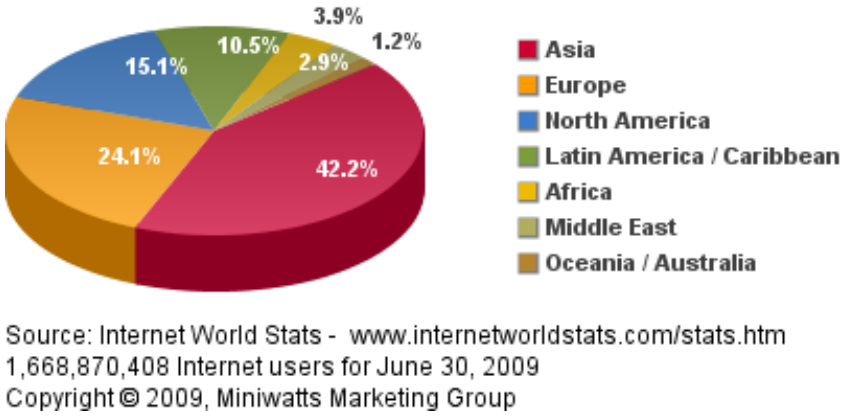


Fig. 2.2 - Internet users by world regions

Also, the "Internet Systems Consortium"[13] shows that in January of 2009 more than 600 million hosts were identified on the Internet, as it can be seen in figure 2.3:

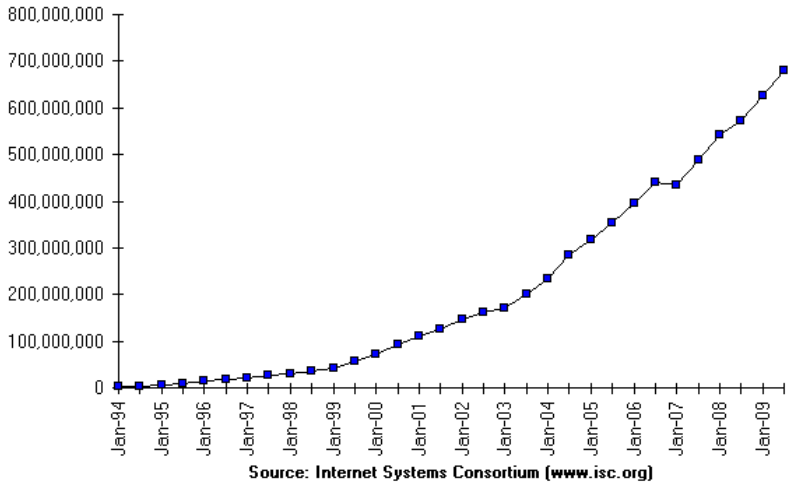


Fig. 2.3 - Internet domain survey host count

By seeing these numbers, one can say that "the Internet is a big place". The Internet has also some properties that make its characterization a very difficult task. Some of these properties are:

- Heterogeneity. The Internet is a heterogeneous system. The internet topology is constantly changing and the routes are almost always asymmetric.
- Scaling. The Internet growing rate is very fast.
- Daily activity patterns. The activity inside the Internet does not obey to a regular behaviour; it has been recognized that the activity inside the Internet obeys a daily pattern where the major part of the activity occurs during day time and it decreases during the night.

Another problem that could be found when simulating something as big as the Internet would be the requirements in terms of necessary equipment for the simulation.

With all this in mind, we can see that is impossible to simulate the whole Internet, but we can simulate a portion of it, taking into consideration actual and typical network scenarios (because we cannot restrict ourselves to one typical scenario, we must consider a spectrum of scenarios), protocols and studies referring the normal behaviour of the Internet users. In this way, we can simulate (or in the case of this work, emulate) an environment that will allow obtaining data that can be used to analyze the behaviour of the system.

2.4. Simulators & Emulators

Simulators and Emulators are a very effective and cheap way to create environments that otherwise could be very expensive to deploy with real equipment. In terms of computer networks, these tools can have multiple networked computers, routers and data links. They are also useful to test new networking protocols or to change existing ones in a controlled and reproducible manner saving a lot of time and money.

Currently we can find a huge variety of simulators and emulators, from payware to freeware solutions, both for educational and research purposes. This sub-section will present a list of some of the most well known simulators and emulators that can be found, as well as some of the reasons for using an emulator in this dissertation.

2.4.1. Simulators

A general definition that can be found is that a simulator is a system that can imitate conditions or characteristics of a real process or in this case, a real network scenario. These tools typically run on a single computer in order to produce a statistical analysis of the environment that is being simulated.

In a simulator, the behaviour of the network can be modelled in two different ways:

- Using mathematical formulas like logarithmic distributions or Markov models[1] in order to calculate the interaction between network entities involved in the simulation.
- Capturing network flows from a real network and playing them back in the simulated network in order to see how it behaves.

[1]Markov Models are used for analyzing complex systems. The name derives from the Markov property that states that given the current state of the system, the future evolution of the system is independent of its history.

Depending on the method of simulation, simulators can be divided into two groups:

- Discrete simulators or discrete event simulators. These simulators use discrete event simulations to model the network. In these simulations, the network is modelled using a set of variables and they have a number of states. An event occurs when the value of the variable changes and this is represented in the system's states.
- Continuous simulators. They are based in the continuous simulation method in order to model the network. Like the previous method, continuous simulation also has state variables; the difference between them is that, in this method, the state variables change continuously with time.

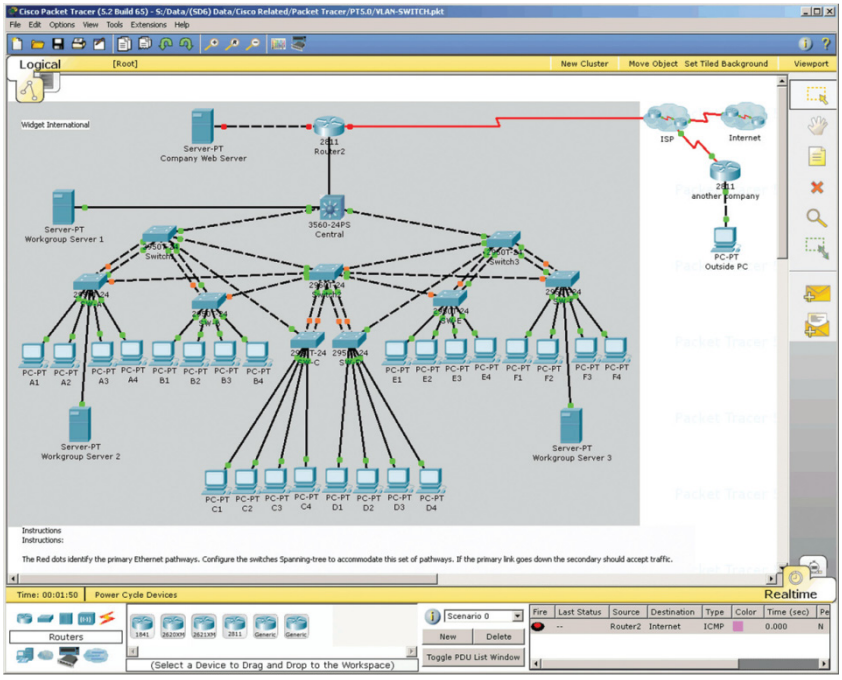
There is a vast list of simulators that can be found today; the list presented below shows some of the most well known network simulators.

2.4.1.1. Packet Tracer

Packet Tracer [14] is a multi-platform network simulator developed by Cisco for the "Cisco Networking Academy" program. It is intended for educational purposes (to obtain Cisco Certifications) and it supports the majority of the protocols needed to obtain the CCNA and CCNP certifications. The Packet Tracer has a user-friendly GUI and, for educational purposes, it has the advantage of not needing real network equipment because it also simulates the way to make connections between the equipments, the topology to study, the equipments and the peripherals to simulate.

The Packet Tracer simulator also has the possibility of "pause and resume" the virtual network flows created inside the simulator at any time and see the path and content of every packet in the network so that the students can analyze the packets at any point in the topology. For research purposes, this simulator is not used because it lacks the ability to generate real traffic and because the program was developed only for educational reasons in order to help students to better understand how computer networks work.

Figure 2.4 shows an example of the Packet Tracer workspace, having also a network topology already designed in it.



Source: Packet Tracer datasheet - http://www.cisco.com/web/learning/netacad/course_catalog/docs/Cisco_PacketTracer_DS.pdf

Fig. 2.4 - Packet Tracer workspace

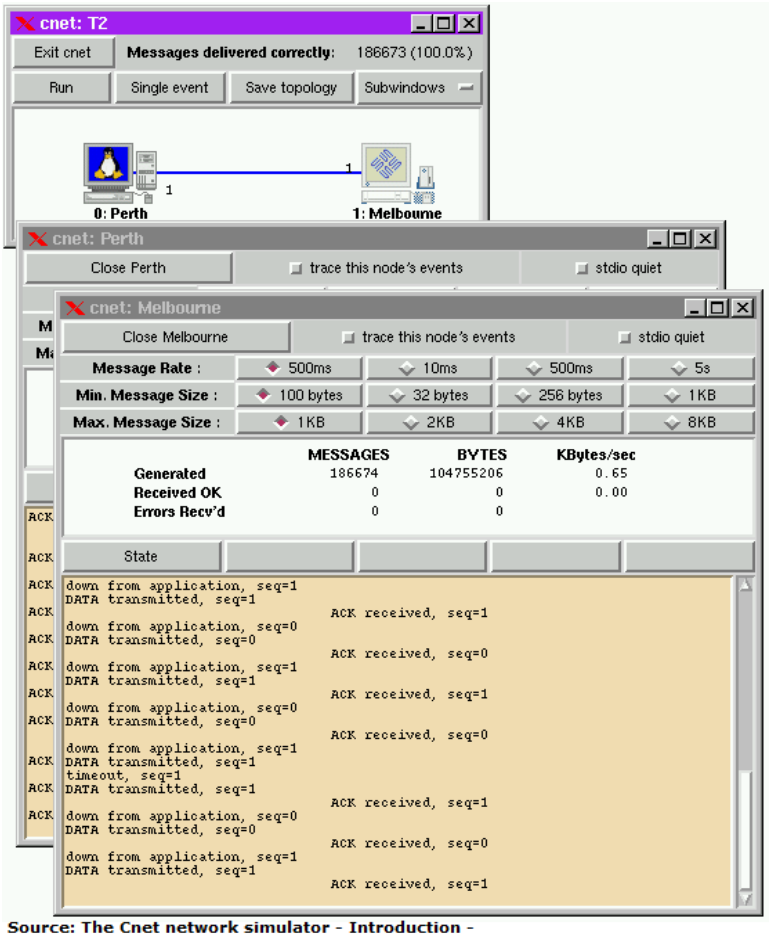
2.4.1.2. Cnet

Cnet [15] was developed by Chris McDonald at the University of Western Australia. The Cnet simulator is a discrete-event simulator that runs on Unix systems. Cnet allows simulating a network with various data-link layer, network layer, routing and transport layer networking protocols. The simulations in Cnet can scale to a few hundred nodes.

This simulator can display the simulated network in two different ways:

- In an ASCII terminal or
- In "Tool Command Language" (Tcl/Tk). In this option, the Cnet provides a graphical interface with the representation of the network where the parameters can be changed while the simulation is running.

Examples of the graphical interface of this simulator and the topology file in Cnet can be seen in figures 2.5 and 2.6, respectively.



Source: The Cnet network simulator - Introduction - http://www.csse.uwa.edu.au/cnet/introduction.html

Fig. 2.5 - Cnet graphical interface

```
/* A simple 2 node point-to-point network topology */

compile          = "stopandwait.c"

bgimage          = "australia1.gif"
drawframes       = true

messagerate      = 500ms,
propagationdelay = 700ms,
probframecorrupt = 3,

host Perth {
    ostype  = "palm"
    x=100, y=100
    messagerate  = 1000ms,

    link to Melbourne
}

host Melbourne {
    ostype  = "linux"
    east of Perth

    link to Perth
        { probframeloss = 2 }
}
```

Source: The Cnet network simulator - Topology files - http://www.csse.uwa.edu.au/cnet/topology.html

Fig. 2.6 - Cnet topology file

2.4.1.3. GTNets

GTNetS [16] was developed by Dr. George Riley. The "Georgia Tech Network Simulator" is an open-source network simulation environment for windows and UNIX systems made in C++. The GTNetS simulator allows the design of large scale networks and the study of the network behaviour under a variety of conditions or situations.

From the GTNetS documentation [17] the following quotation can be taken: "*The design philosophy of GTNetS is to create a simulation environment that is structured much like actual networks are structured*". In other words, the main objective of the GTNetS is to simulate networks as real as possible and this is achieved by the way GTNetS handles the situations inside the simulation environment. For example, each of the objects inside the simulation that represents a node in the network can have more than one interface and each interface has an associated IP address.

Some of the main features and capabilities of the GTNetS simulator are shown below:

- Connection oriented and connectionless applications.
- Support for a variety of TCP models.
- Layer 2 and Layer 3 support.
- Routes can be statically or manually calculated by the user.
- Supports for packet tracing and statistics gathering inside the simulation.

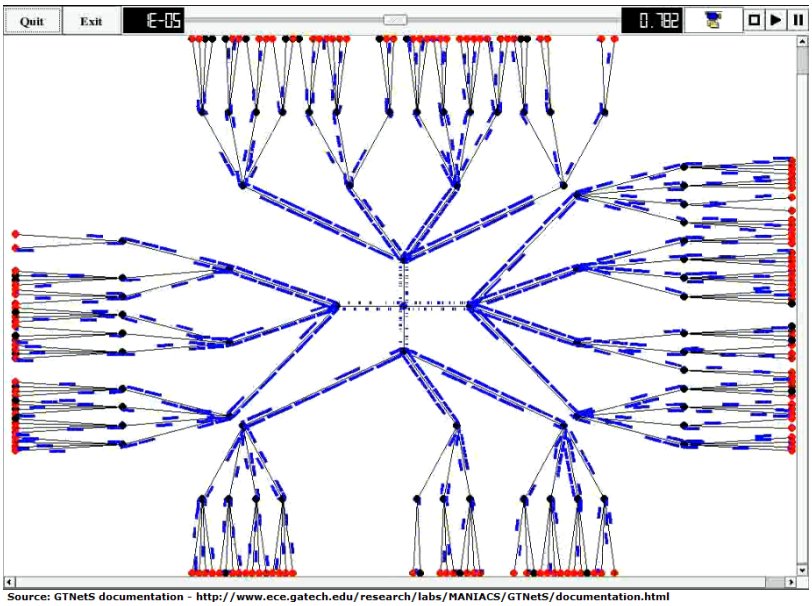
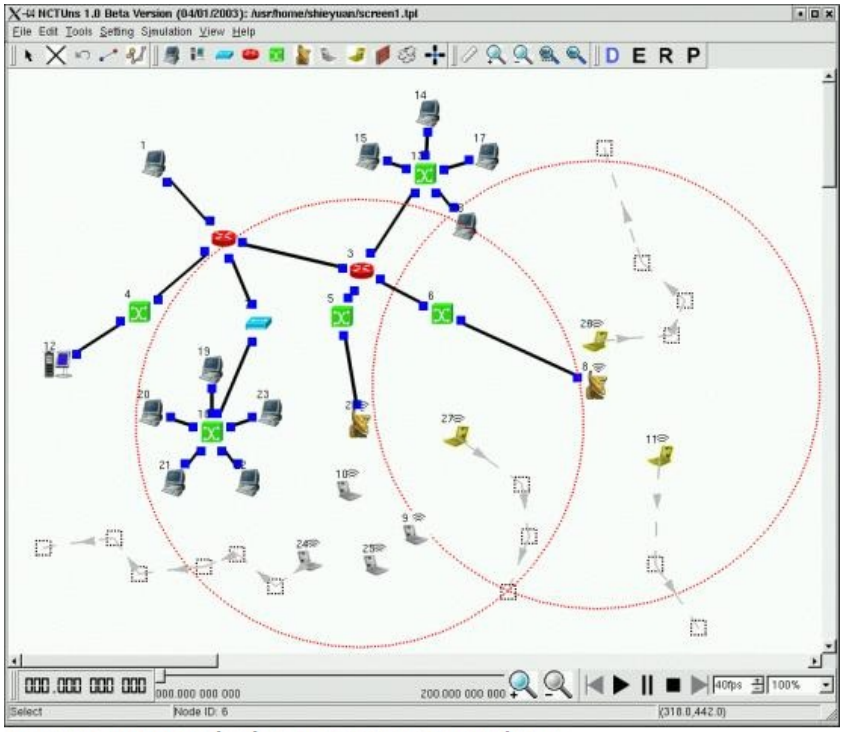


Fig. 2.7 - Animated simulation in GTNetS

2.4.1.4. NCTUns

NCTUns [18] was developed by Professor S. Y. Wang from the National Chiao Tung University. The NCTU Network Simulator is a TCP/IP simulator programmed in C++and one of its main features is that it is also an emulator. The core technology behind the NCTUns was developed by Professor Wang himself during its PhD studies in the Harvard University. The principal characteristic of the NCTUns simulator is that the network topology is built combining tunnel devices and link simulations. The simulations are more realistic in this environment because the applications and the protocol stack are not modified in the simulator.

The NCTUns supports a variety of important networks (Ethernet based IP networks, wireless LAN networks and GPRS cellular networks are some of them). It also supports various networks devices (like router, hubs and switches) and protocols (like OSPF, UDP, HTTP, FTP).



Source: SIMREAL Technology Homepage - Screenshots - http://nsl10.csie.nctu.edu.tw/

Fig. 2.8 - NCTUns topology editor

2.4.1.5. The Network Simulator – NS2

NS2 [19] was developed by the Information Science Institute (ISI) at the University of Southern California (USC). NS2 is the second version of NS and is one of the most popular open source network simulators.

The NS2 simulator is a discrete event driven network simulator programmed in C++ and OTcl (Tcl script language with object-oriented extensions developed at MIT) and the project is supported through DARPA. It provides support for TCP, routing and multicast protocols over wired and wireless networks simulation and many of its packages have been developed by non-benefit groups.

One characteristic of the NS2 is that, in order to reduce packet and event processing time, the event scheduler and the basic network objects are written and compiled in C++.

The NS2 also has the capability of becoming an emulator. This is done using a series of tap agents and network objects. The tap agents can insert live network data into the simulation and extract live data from the simulation and the network objects provide an entry point for live data.

To visualize the NS2 topologies, the simulator supports the use of **NAM: Network Animator**. The NAM is an animation tool used to visualize network simulation and real world traces. Figure 2.9 shows the representation of a topology made in NS2 in the NAM animator.

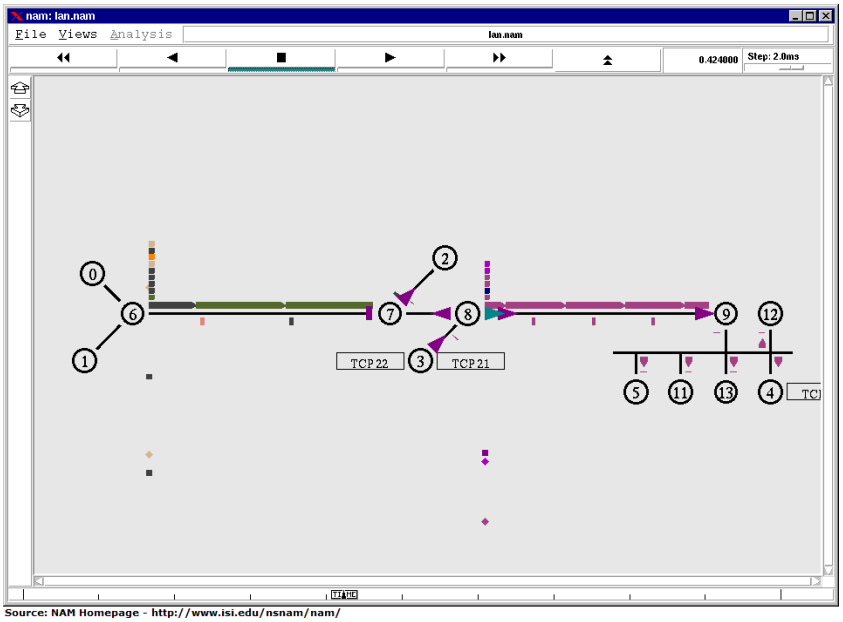
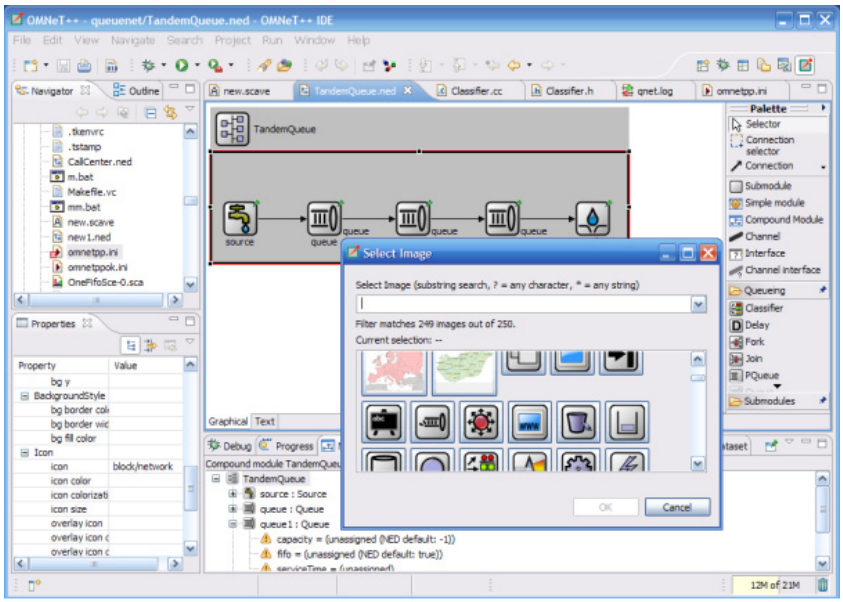


Fig. 2.9 - Topology in NAM

2.4.1.6. OMNeT++

OMNeT++ [20] is a multi-platform, discrete event simulator programmed in C++ and developed by András Varga. The components inside the simulator are assembled into models using the "NEtwork Description Language" (NED) of OMNeT++. An example of the NED editor can be seen in figure 2.10.

The OMNeT++ can be used for modelling wired and wireless networks, queuing networks and network protocols. Because its open-source and extensible, this simulator is very popular for academic purposes.



Source: OMNeT++ Homepage - Screenshots - http://www.omnetpp.org/index.php/home/what-is-omnet/3442

Fig. 2.10 - NED topology editor

2.4.1.7. OPNET

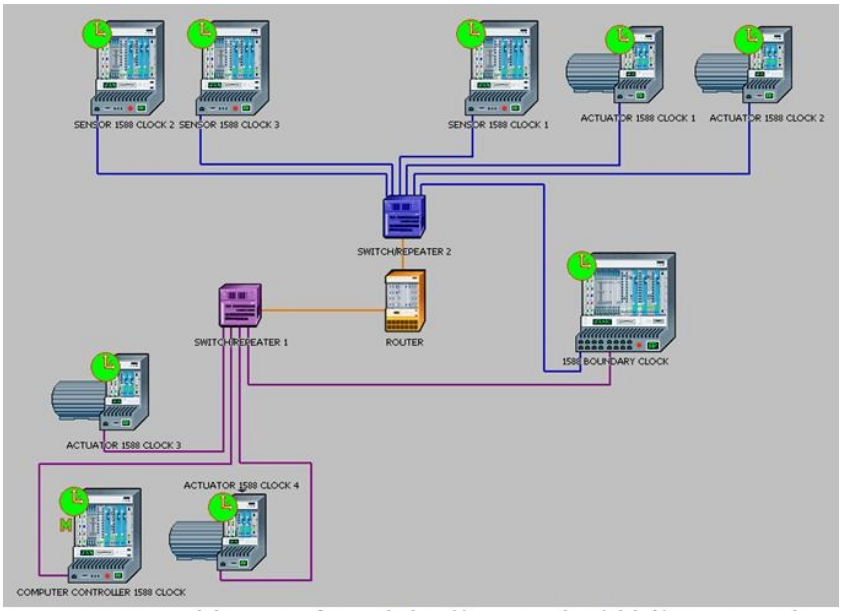
OPNET [21] is an object oriented, discrete event simulator developed by OPNET Technologies Inc. The OPNET simulator has three main functions: modelling, simulation and analysis.

According to the OPNET documentation, some of the main features are:

- Fast discrete event simulation engine.
- Object-oriented modelling.
- Hierarchical modelling environment.
- Customizable wireless modelling.
- Discrete event, hybrid and analytical simulation.

- Grid computing support.

These and other features allow the users of this simulator to design and study different kinds of networks, protocols, devices and applications. Figure 2.11 shows an example of an OPNET network scenario.



Source: OPNET Research (University of Brescia) - http://www.ing.unibs.it/~label/OPNET_Brescia.htm

Fig. 2.11 - OPNET network scenario

2.4.2. Emulators

In terms of computer networks, network emulators work in a similar way as simulators. Their main differences are that network emulations run in real time while network simulations do not. In network emulators, the network behaves like real networks in order to see, measure and test their performance. Real equipment can be connected to the emulation (computers, routers and other network entities) and they will behave exactly as if they were in a real network.

In terms of network emulators, several solutions can be found, from emulation software that can be connected to other emulation environments to form a larger network topology to online emulation testbeds with real computers and resources in order to obtain the results of the designed topologies.

The following list shows some of the networks emulators that are used nowadays.

2.4.2.1. Dummynet

Dummynet [22] was developed by Luigi Rizzo at the University of Pisa. The Dummynet emulator is a link emulator that was originally developed for the test of network protocols and has become a very popular tool because of its capabilities and features and it is also one of the core components of the Emulab testbed. The dummynet emulator's main platform is FreeBSD but it can also be used in MAC OS X and recently it has also become available for Linux and OpenWRT.

Dummynet works by capturing packets using ipfw rules (ipfw is one of the FreeBSD firewalls) when they pass in the protocol stack. When captured, the packets will pass through one or more objects called "queues" and "pipes". These objects can simulate the effects of bandwidth limitations, delays, packet losses, bounded-size queues and multipath.

The usage of the link emulation technique in dummynet is related with the fact that this technique is very similar to traffic shaping; a feature used in several systems and routers to guarantee service limitations. In the latest update, the dummynet received some new features. It now includes various queue management schemes like FIFO, RED and WF^2Q++ and support for MAC layer emulation was also added.

Figure 2.12 shows where dummynet captures the packets (according to the developer).

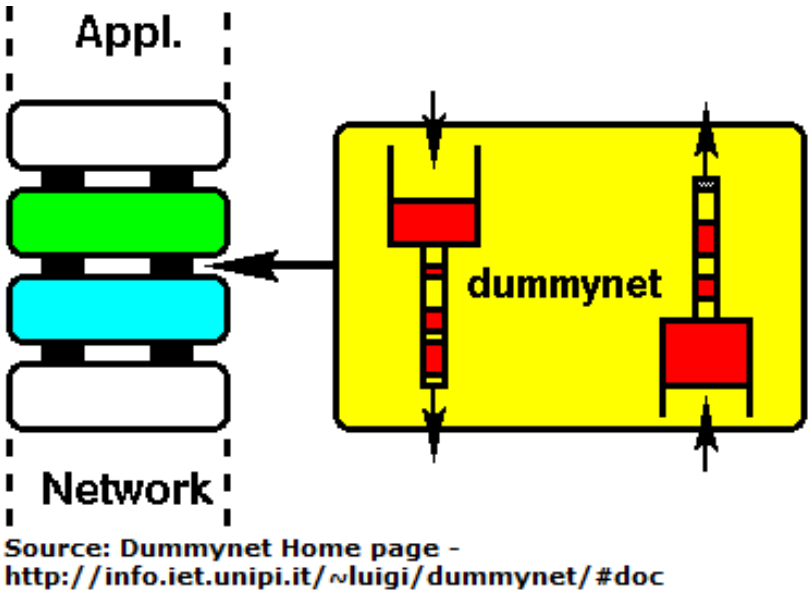


Fig. 2.12 - Dummynet packet capture

2.4.2.2. NetEm

NetEm[23] is supported by the Linux Foundation. The NetEm can emulate the properties of Wide Area Networks (WAN) in order to test network protocols. The NetEm is built using Quality of Service (QoS) and Differentiated Services (diffserv) modules that can be found in the Linux kernel and is controlled by the command line tool "tc", which is part of the "iproute2" package of tools. The communication between the command and the kernel is done via the Netlink socket interface and the requests are encoded into a standard message format that will be decoded by the kernel.

A GUI also has been developed in order to run the commands used in NetEm.

2.4.2.3. SIMENA

The **SIMENA**[24] solutions developed by SIMENA® involve traffic generation, packet flow switch, network taps and network emulator. In terms of the network emulation, the SIMENA solution works in the Ethernet layer so it does not require any change in the network. The emulator can be used for any network protocol (IP, IPX, etc).

The datasheet that is located on the SIMENA webpage mentions the following characteristics for this emulator:

- Bidirectional emulation
- Unidirectional emulation
- Simultaneous emulations
- Layer 2 & Layer 3
- IPTV MPEG impairments
- Latency, jitter
- Accumulate & burst
- Packet loss
- Bandwidth throttling
- Duplicate packet
- Out of order packet
- Congestion
- Carrier loss
- Queue size
- VLAN, stacked VLAN
- MPLS, stacked MPLS
- Fragmentation
- BER

- Jumbo frame
- DiffServ
- Queuing
- Mesh emulations
- G.1050/TIA-921

An example of the SIMENA emulator is shown in figure 2.13

Source: SIMENA Datasheet -
http://www.simena.net/network_emulator_datasheet.pdf

Fig. 2.13 - SIMENA emulator

2.4.2.4. Emulab

Emulab [25] is a network testbed where researchers can develop, test and analyze their networks topologies in a controllable environment. This testbed manages a PC cluster that can provide a space and time-shared facility for the study of network topologies and where it is also possible to run the operating system that is desired by the researcher. The emulab testbed works with the NS-2 emulation option so that simulated networks can interact with real networks and also with the network resources.

As shown in figure 2.14, emulab can be deployed in sites all over the world but the primary installation of the testbed is encountered at the University of Utah and is run by the "Flux Group", that is a part of the School of Computing.

Emulab is available without any charge to most researchers but in order to qualify for the usage of the testbed, it is necessary to apply with a new project and be

evaluated. The policy documents needed for this process can be found on the Emulab website.

Once the project is approved and the specifications are in the database, the emulab will attempt to map the virtual topology in the cluster in order to meet the needed resources.

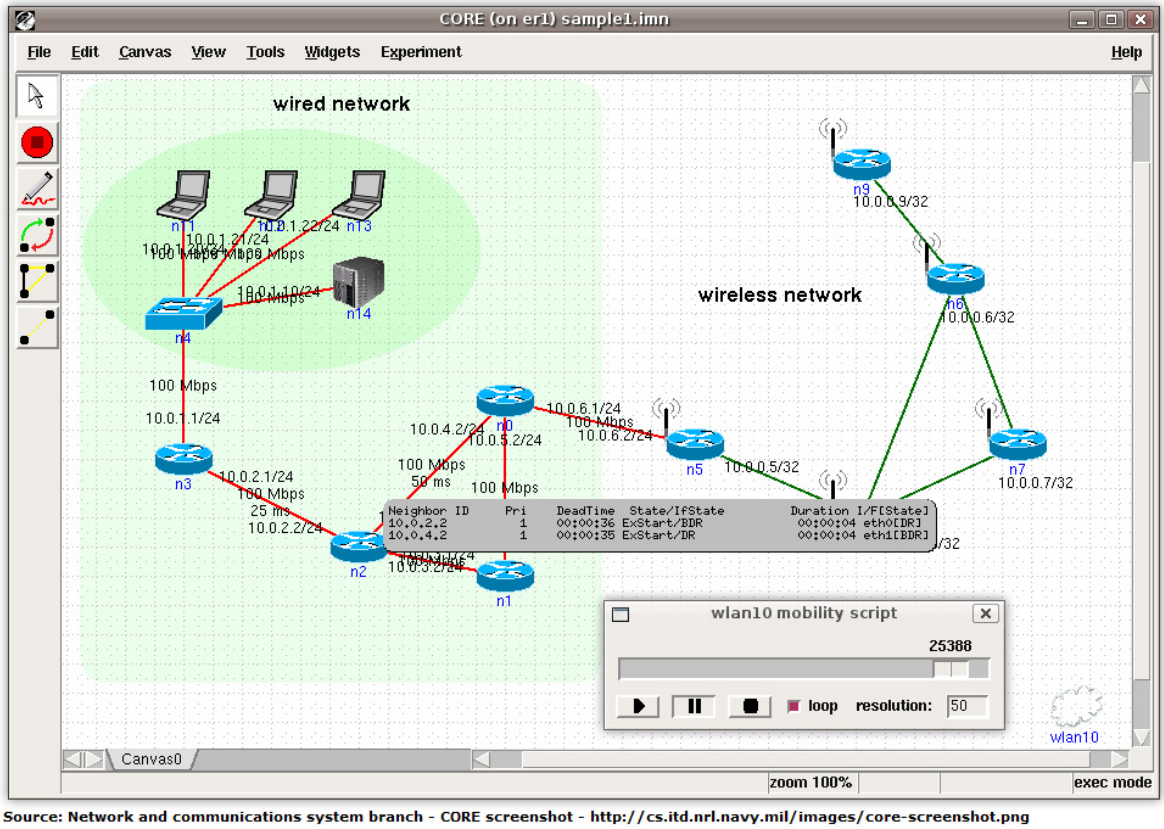
Source: Emulab website - http://users.emulab.net/trac/emulab/wiki/OtherEmulabs

Fig. 2.14 - Emulab sites around the world

2.4.2.5. CORE

The **CORE** [26] (Common Open Research Emulator) is a network emulator developed at the Naval Research Laboratory in Washington D.C that works on the FreeBSD platform. This emulator presents a hybrid approach in terms of simulation tools and dedicated testbeds by allowing the emulation of routers and other hosts using virtualization and simulating the links between them. The main advantage of this approach is that it can run applications through the emulated network without the expensive requirements of having the real equipment.

The GUI used in CORE is scripted in Tcl/Tk language and an example of this GUI is shown in figure 2.15.



Source: Network and communications system branch - CORE screenshot - http://cs.itd.nrl.navy.mil/images/core-screenshot.png

Fig. 2.15 - CORE GUI

2.4.2.6. Dynamips

Dynamips [27] was developed by Christophe Fillot. The dynamips emulator is a text-d-based, open-source tool for the emulation of Cisco routers. It began as a project to only emulate the 7200 router series; nowadays it can emulate several Cisco platforms (1700, 3600, 3700 and 2600 series) and has a huge community of users. Useful for training especially for the Cisco certifications, it also has a great potential in the research area. Dynamips has the ability to read actual IOS images of the Cisco routers in order to emulate their functionality and it also allows the use of real and virtual host connected to the emulation environment in order to create the interaction between the emulation and the real environment.

In order to simplify the design of virtual networks in Dynamips, the Dynagen tool was created. The dynagen[28] was developed by Greg Anuzelli and is written in python, which gives the advantage of being usable in any platform that has a python interpreter.

According to the Dynagen tutorial [32], Dynagen simplifies many things when using dynamips:

- Simple syntax for the specification of the router configuration.
- Simple syntax for interconnecting routers, bridges, frame-relay and ATM, and Ethernet switches.
- Can work in Client/Server mode.
- Provides a management CLI for listing devices and for the operation of those devices.

2.4.3. Emulators vs simulators

All the previously presented tools have their advantages and disadvantages. Based on a series of studies and papers[6, 33], the choice for this dissertation was the use of an emulator because in simulators the response of the network and the interaction of the network agents is mainly calculated using mathematical formulas. This is good for testing new implementations of protocols or new topologies in order to find optimal results. In emulators the interaction of the network agents with the emulated network is made exactly as they were in real networks: the performance of the agents and the emulated network when using all sorts of applications and their reactions to changes is similar to what happens in the real world, so it can be used as a comparison term.

Network emulation can combine real elements such as end-hosts and protocols with simulated elements, like network link and background traffic; in pure simulators that combination cannot be done.

Network emulation runs in a real-time environment while simulation usually run in a virtual-time environment (the concept of time is usually not necessary in the implementation inside the simulation).

As said earlier in this chapter, the Internet is always evolving and is a very heterogeneous system that cannot be represented with the simplified models usually used by some simulators.

The main advantage of using an emulator for this work is given in the network security area. As previously said, one of the main objectives of this dissertation is to show a real-time environment where we can obtain real traces, without having to concern with some issues previously mentioned about the usage of public traces, and to create a platform for the evaluation of new methodologies for the detection of

security attacks. In an emulator environment we can achieve a higher confidence degree in our results and in this kind of environment detection methodologies or applications can be tested in order to analyse their performance.

2.5. Summary

This chapter presented a definition for Ground Truth, its characteristics and requirements; then, the importance of Ground Truth in the network security area was discussed, as well as the problems of using public traces in order to test new methodologies for attack detection and the difficulties of simulating the Internet.

We also presented some brief definitions of simulators and emulators. Moreover, a brief list of some simulators and emulators was also shown, together with their main characteristics and types as well as the main reasons for using an emulator in this work.

3. GNS3

3.1. What is GNS3?

The **GNS3** [29] is a multi-platform, open-source Graphical Network Emulator primarily developed by Jeremy Grossman. The GNS3 allows the emulation of network topologies by emulating the Internetwork Operating System of the Cisco routers with the help of Dynamips and Dynagen (already discussed in the previous chapter).

As explained in the previous chapter, the core program behind the emulation process is called Dynamips, and the Dynagen tool runs on top of it to create a user-friendly, text-based environment. The GNS3 provides the graphical front-end for Dynagen so that users can create the topologies in a graphical and user-friendly environment.

Figure 3.1 shows an example of what a GNS3 workspace looks like.

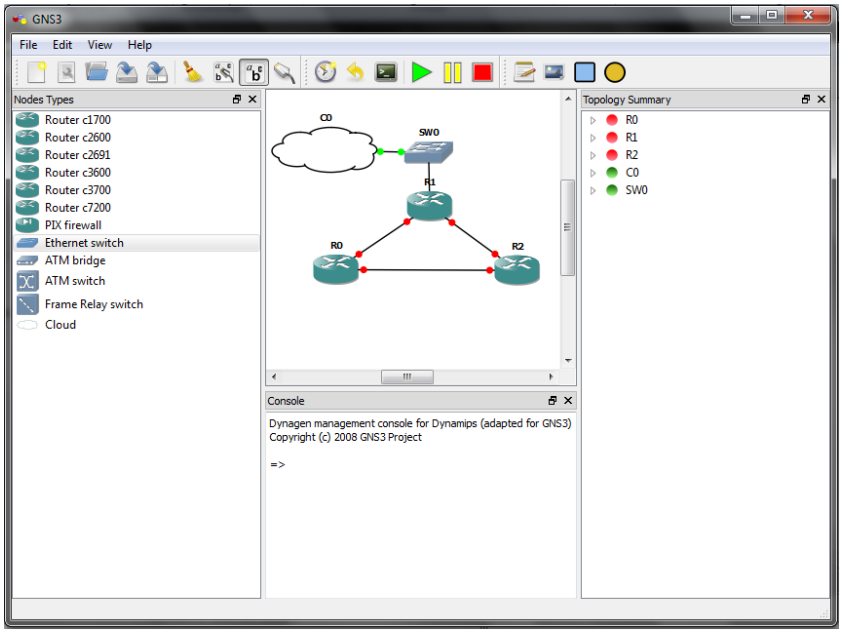


Fig. 3.1 - GNS3 Workspace

The GNS3 environment is divided into four main spaces:

- Node type section;
- Console section;
- Topology Summary section
- Topology work area.

In the node type section there is a list of all available nodes in GNS3 (Routers, Ethernet switch, ATM bridge, ATM switch, Frame Relay switch and a cloud). In order to use some of these nodes (the routers, for example), the user must have the IOS image associated with the node to emulate it. A complete list of the IOSs supported by GNS3 is given in table 3.1 (taken from the tutorial found in the GNS3 website [30]).

List of supported Cisco IOS		
1710	2611	2691
1720	2611XM	3620
1721	2620	3640
1750	2620XM	3660
1751	2621	3725
1760	2621XM	3745
2610	2650XM	7200
2610XM	2651XM	

Table 3.1 - Cisco IOSs supported by GNS3

The console section shows the Dynagen commands and allows the direct input of those commands instead of the graphical commands provided by GNS3.

The Topology summary section shows all the nodes involved in the emulation and how are they connected to the other nodes (shows the node's interfaces where the link is made).

The topology work area is where the design of the network topologies occurs. All the items found in the Node type section can be dragged into the Topology work area. Once there, the user can access all the options given by GNS3 to that specific node (in the case of routers, for example, the user can choose the type of interfaces he wants for every specific router in the topology).

Because of the usage of real IOS images, the GNS3 can consume the PC resources very fast even when the nodes are not processing traffic. In order to reduce and optimize the PC resources used by GNS3, when the routers are placed in the topology, the user needs to calculate the "IDLE PC" value for all the routers. This value is calculated after starting the router and it can be found on the router menu (right-click with the mouse on top of the desired router), as shown in figure 3.2

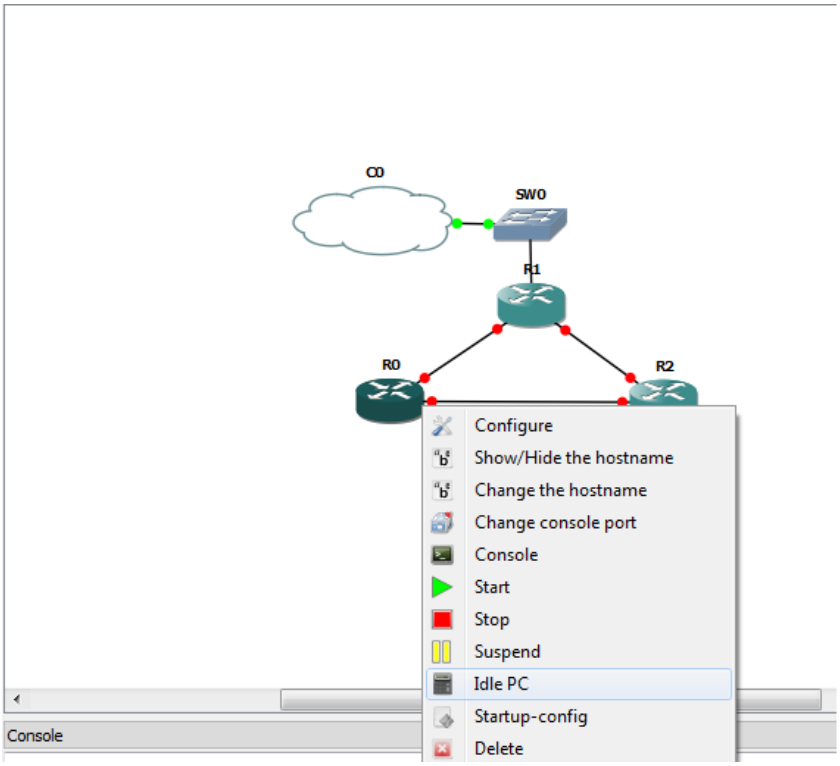


Fig. 3.2 - Idle PC option

When the "Idle PC" value is associated, the resources of the PC are only used when needed, and in that way the user can optimize the resources and can have more complex topologies in the same computer or have other programs running at the same time.

3.2. Topology Files

The topology files used by GNS3 (.net files) have the same syntax as the dynagen files; they are text-based files that can be opened with any text editor and very easy to understand. An example of a GNS3 topology file can be viewed in the following figure:

example.net - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
autostart = False
[localhost:7200]
    workingdir = C:\Users\Carlos\AppData\Local\Temp
    udp = 10000
    [[3745]]
        image = C:\Program Files\GNS3\c3745-advipservicesk9-m.124-4.T1.bin
        ghostios = True
    [[ETHSW SW0]]
        1 = access 1
        2 = access 1 nio_gen_eth:\device\npf_{04d8ce57-1f85-4d48-9ef7-c845bf80db93}
        x = -54.186291501
        y = -132.828427125
    [[ROUTER R0]]
        model = 3745
        console = 2000
        f0/0 = R1 f0/0
        f0/1 = R2 f1/0
        slot1 = NM-1FE-TX
        x = -186.0
        y = 31.0
    [[ROUTER R1]]
        model = 3745
        console = 2001
        f0/0 = R0 f0/0
        f0/1 = R2 f0/1
        slot1 = NM-1FE-TX
        f1/0 = SW0 1
        x = -50.0
        y = -61.0
    [[ROUTER R2]]
        model = 3745
        console = 2002
        f0/1 = R1 f0/1
        slot1 = NM-1FE-TX
        f1/0 = R0 f0/1
        x = 77.0
        y = 32.0
[GNS3-DATA]
    m11 = 0.707106781187
    m22 = 0.707106781187
    [[Cloud C0]]
        x = -248.375108319
        y = -154.928932188
        connections = SW0:2:nio_gen_eth:\device\npf_{04d8ce57-1f85-4d48-9ef7-c845bf80db93}
```

Fig. 3.3 - Content of a GNS3 topology file

As can be seen from the previous figure, the nodes and IOS images used in the simulation are inside double brackets ([[ ]]). Inside the image section we can find the path of the image that is used in the emulation. Inside the sections of the routers nodes we can find the models of the routers to be emulated, the type of slots configured in the routers, the connections that were made for every interface, the coordinates inside the GNS3 work area and the console. This console shows the TCP port used by each router in the emulation (the connections via TELNET to connect to the routers are made through these ports).

The GNS3 emulator also gives the possibility of connecting virtual PCs in order to test the basic functionality of the emulated network. This is done with the help of the Virtual PC Simulator (VPCS) that can be found in the download section of the GNS3 website (http://www.gns3.net/download) or in the following site: http://www.freecode.com.cn/doku.php?id=wiki:vpcs .

The VPCS is a PC simulator currently in version 0.16c and it is capable of simulating up to 9 PCs to test the basic functionality of the network. The PCs simulated by this tool can ping and traceroute every other host or routers found in the emulation and can also obtain their IP address via DHCP when routers are configured as DHCP servers.

As previously said, the VPCS gives us basic functionality to test an emulated network, but for this work it is not enough. The GNS3 also gives the possibility of connecting the network to a real network, to real network equipment and hosts or to virtual interfaces that can act exactly as PCs making any kind of requests to a server on the network. Of course, there are some important issues to take care when connecting the emulated environment to the "real world". Those aspects will be explained in the next sub-section.

3.3. Connecting the GNS3 environment to real and virtual equipment

In the scenarios that will be tested in this work there is a need to emulate the effects of a real network or, in other words, have end-hosts that make requests to a server (HTTP requests, FTP requests and mail services) and VPCS has no potential to perform this kind of operations. The solution that was found to create the test platform was a combination of real and virtual equipment, making the host PC (PC where the emulator is running) act as a bridge in order to connect the real devices to the emulation. In a Linux platform there are two utilities that, when combined, allow this to be done: the Bridge and the TUN/TAP utilities, both of them can be found in the repositories of the Linux platforms.

3.3.1. Bridge

A bridge allows the interconnection between LANs within the same protocol (in this case, Ethernet) and it works at the data-link level of the network. To do this in GNS3, a virtual interface inside the emulation has to be associated with a real interface in the PC (Ethernet card) in order to allow the flow of packets to travel between the real equipment and the emulated environment.

To create a bridge interface in Linux, the followings commands are used in a Linux terminal:

brctl addbr br0
ip l s dev br0 up

The first command creates the bridge interface "br0" (br0 is the name given to the bridge) and the second command activates the device so that it can be used. After creating the bridge, the real and virtual interfaces (Ethernet card and TAP interface, respectively) are associated to the bridge using the following commands:

brctl addif br0 tap1
brctl addif br0 eth0

The command *"brctl addif"* shown here adds the interfaces needed to create the bridge. In this particular case (connecting the emulated network to real equipment), it is very important to set the interfaces associated to the bridge in "Promiscuous mode". The "Promiscuous mode" allows all the data packets from the emulated network and the real equipment to be received and transmitted between the two environments.

3.3.2. TUN/TAP Interface

The TUN/TAP interfaces are part of the "uml-utilities" package found in the Linux repositories. These virtual interfaces can implement network devices via software. In other words, these virtual interfaces can behave as real network adapters and their main difference is that the TAP interfaces operate with layer 2 packets while TUN interfaces operate with layer 3 packets. For this work, the virtual interface has to be able to be in a bridge and also to do switching, so TAP interfaces were chosen.

In order to use a TAP interface, the user needs to add the TUN module with the following command:

modprobe tun

The *"modprobe"* command, as explained in the Linux man pages, can add a module to the Linux kernel. In this case, the added module is the module responsible for the creation of TUN/TAP interfaces.

After adding the module, the followings commands are used to create and activate the TAP interface:

```
tunctl -t tap1
ip l s dev tap1 up
```

As also explained in the Linux man page, the *"tunctl"* command allows the creation of a persistent TUN/TAP interface and the *"-t"* parameter is used to give the interface name (in the case above, tap1). The second command, as explained in the bridge sub-section, allows the activation of the interface.

As explained before, in this work the TAP interfaces where used for two reasons: the first reason is the creation of a bridge that allows the connection between the emulation and the real equipment and the second reason is to emulate end-hosts inside the emulated network. These hosts need to be capable of sending a variety of requests to a server, in the same way as a real user normally behaves.

For the second reason, the usage of several TAP interfaces inside the emulation is a process that requires taking into consideration a slightly different approach when configuring the Linux routing table so that all the virtual interfaces can communicate with the real equipment. This process will be explained in the next chapter.

The configuration of these virtual devices in this work can be seen in the Appendix C.1.

3.4. Summary

This chapter explained the emulation software GNS3 that will be used in this work, its main characteristics and the virtual devices that are needed to connect an emulated scenario inside GNS3 to real equipment.

4. Emulation

Before explaining the Scenarios used in this work, it is necessary to explain all the necessary configurations that were made in the servers, the routers and the PCs themselves in order to configure all the implemented protocols and services. The following sub-sections of this chapter will explain all the configuration process.

4.1. Server Configuration

In order to configure all services without problems, all the modifications to the server have to be made using root privileges.

4.1.1. Interface configuration

The Network Interface Card (NIC) of the server is configured with one real IP address and several Virtual IP Addresses (VIP/VIPA) so that the DNS will be associated to the real IP address and the first VIPA will be associated with the domain configured for the APACHE, ProFTPD and POSTFIX servers. The other VIPAs are used for testing part of the attacks made in the emulated scenarios (port scans in multiple IPs).

All the IP addresses of the server are static. To configure them, the *interfaces* file, found in the "/etc/network/" directory has to be edited with the desired IP addresses and the networking services of the server have to be restarted in order for the modifications to take effect.

To restart the networking services, the following command is used:

```
sudo /etc/init.d/networking restart
```

The modified *interfaces* file is shown in Appendix A.1.

4.1.2. APACHE

APACHE [10] is a project from the Apache Software Foundation and is the number one HTTP server on the Internet. It is an effort to develop and maintain an open source HTTP server for modern operatives systems. It provides security, efficiency and extensibility. In order to answer to the HTTP requests that are made by the clients, the server is listening on port 80.

The APACHE installation package can be found in the Linux repositories, along with the dependencies. For this work, in order to enable PHP and MySQL support to the server, the following packages were also installed:

- Apache2
- MySQL Client
- MySQL Server
- PHP 5
- PHP 5 - GD
- PHP 5 - MySQL
- PHPMyAdmin

The APACHE configuration file can be found in *"/etc/apache2/sites-enabled"* has the name *000-default*. In this file, all the main options for the APACHE server are configured, like the directory where the websites are stored, the port used to listen for HTTP and the type of log files to be created by the server. For this work, the only modification made to the *000-default* file was the "allowOverride" option (set to "All") under the <*Directory /var/www*> section, as it can be seen in Appendix A.2.

The websites used for this work are stored in the default document root for APACHE (*/var/www/).*

These websites are copies of some news Internet sites (CCN, BBC News, SKY News). The reason behind the usage of these types of Internet sites is because they present a mixture of text and words so that they can be considered as "heavy network traffic", and can also be seen in Appendix A.2.

4.1.2.1. Hyper-Text Transport Protocol (HTTP)

The HTTP [34, 35] is an application protocol that runs over TCP connections and defines the interaction between "Web browsers" and "Web servers", as well as the format of the messages.

Each interaction is constituted by two actions:

- The client sends a request message indentifying the file that it wants to receive.
- The server sends a response message that can be negative or positive. If the message is positive, the message received by the client will also have the contents of the file that it requested.

4.1.3. ProFTPD

ProFTPD [36] is a secure and reliable FTP server available for Linux and UNIX operative systems. The ProFTPD listens to requests on port 21.

The reasons behind the usage of the FTP protocol are to emulate the download and upload of small files that are frequently made by users of the University of Aveiro *ARCA* FTP site. The other reason for using the FTP protocol is to emulate snapshot security attacks, as it will be explained later in this chapter.

The ProFTPD package can be found in the Linux repositories, along with all its dependencies.

The configuration file of ProFTPD is called *proftpd.conf* and can be found in the *"/etc/proftpd/"* directory. In this file, we can configure the type of logs to be made by the server, the FTP account to be used, the upload and download directories and their permissions.

The configuration process of ProFTPD is shown in Appendix A.3.

4.1.3.1. File Transfer Protocol (FTP)

The FTP [37] is a file transfer service that runs over TCP. The FTP server uses two ports to create the connection with the client:

- Port 21: Control link.
- Port 20: Data link.

In a FTP session, the user establishes a control link with the server in order to exchange the needed FTP commands and this link is active until the session is finished. Whenever a data transfer is needed, the server establishes a data link with the client to a port that was previously announced by the client using the *PORT* command. The data link is terminated by the server when the data transfer is over.

4.1.4. POSTFIX

POSTFIX [38] was developed by Wietse Venema as an alternative to the Sendmail E-mail server and it attempts to be a fast and secure E-mail server.

Currently, POSTFIX is one of the most popular E-mail servers for UNIX systems. In its list of main features, we can find support for:

- SMTP.
- POP3.
- Maildir and Mailbox format.
- Virtual domains.
- MySQL Database.

All the features mentioned above were used in this work to create, a "Virtual hosts and Virtual domains" E-mail server. The advantage of using this setup relies in the possibility of using multiple users and domains and the fact that it is easier to administrate because, when adding or removing domains/users, we only need to deal with the created MySQL database.

Figure 4.1 shows the "big picture" of the E-mail server configured for this work.

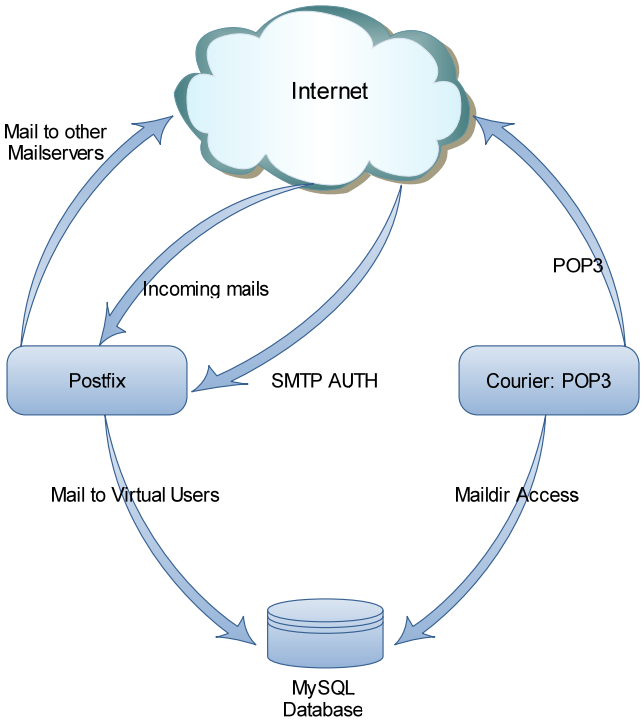


Fig. 4.1 - Interaction between the network, the E-mail server and the database

After installing all the necessary packages found in the Linux repositories (shown in appendix A.4), the following steps are made to correctly configure the POSTFIX E-mail server:

- Create the virtual domain and users MySQL database.
- Configure the Postfix maps. These maps are used by Postfix to know where to look and store the information in the database tables. These maps are:
  - Virtual alias.
  - Virtual domains.
  - Virtual mailbox.
  - Virtual mailbox limit.
  - Relay domains.

  All these maps are created in the "/etc/postfix" directory.
- Create the *vmail* user. The *vmail* user is created in the server machine to store all the e-mails for the virtual host therefore, avoiding the creation of one user account in the server for each e-mail account.
- Configure Postfix to use the maps. This is done by editing the Postfix configuration file "main.cf" that can be found in "/etc/postfix/".
- Configure Postfix to use TLS certificates. Also done by editing the configuration file of Postfix.
- Enable the IMAP authentication by editing the *authdaemonrc* and *authmysql* files found in the "/etc/courier/" directory.
- Enable the SMTP authentication by editing the Postfix configuration file, the *saslauthd* file found in the "/etc/default/" directory and the *smtpd.conf* and *smtp* files found in the "/etc/postfix/sasl/" and "/etc/pam.d/" directories, respectively.
- Create the *postfix* user.
- Restart all the services so that the configurations will take effect.

The detailed process of installing and configuring the Postfix server is shown in Appendix A.4.

4.1.4.1. Postfix Admin

The Postfix Admin [39] is a web based administration interface that allows the management of the virtual domains and users in an easy way. It allows adding or removing virtual domains and users and can be found in the Linux repositories.

Before using it, the *config.inc.php* file found in the "/var/www/postfixadmin/" has to be edited in order to enter the setup menu (the setup menu is accessed via the postfix admin setup URL: http://domain/postfixadmin/setup.php. where "domain" is the previously configured virtual domain).

The configuration file for the postfix admin is shown in the Appendix A.5.

4.1.4.2. Simple Mail Transfer Protocol (SMTP)

SMTP [40, 41] is the standard protocol for outgoing e-mail messages over the Internet and uses the TCP port 25. The last update of this protocol happened in 2008 with the introduction of the Extended Simple Mail Transfer Protocol (ESMTP).

In SMTP, the sender establishes a two-way connection with a receiver, where this receiver can be the final destination of the message or an intermediate host. The SMTP commands are generated by the sender of the message and the SMTP replies are sent from the receiver in response of those commands.

4.1.4.3. POST Office Protocol version 3 (POP3)

POP3 [42] is one of the most used and standard protocols for e-mail retrieval and uses the TCP port 110 for the connections between the client and the server.

The client establishes a TCP connection with the server and authenticates himself. After a successful authentication, the messages are downloaded to the client and erased from the server. The connection is terminated after the download of all the messages in the server and the messages can be read by the client in off-line mode.

4.1.5. MySQL

MySQL [43] is an open-source, multi-tasking and multi-user database server found in the Linux repositories, which uses SQL (Structured Query Language) developed and supported by Sun Microsystems. For this work, MySQL was used to create the virtual domain and virtual user's database linked to Postifx to complete the E-mail server.

For the database modelling, the MySQL Worbench [44] software was used. The MySQL Workbench is an open-source and multi-platform database modelling tool for creating and designing databases.

The model of the database that was created to use with Postfix is shown in the following figure

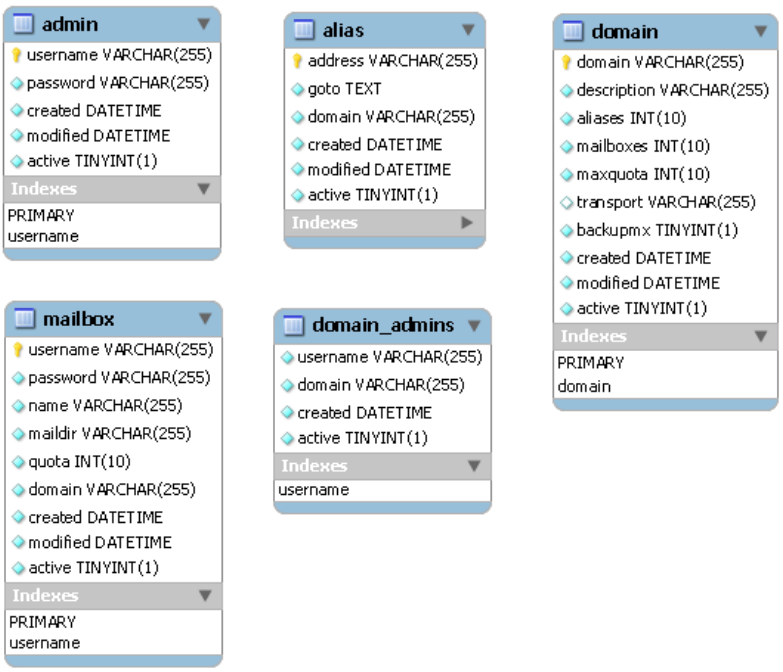


Fig. 4.2 - Database tables created for the E-mail server

The *admin* table stores the information for the general Postfix administrator.

The *Mailbox* table stores the credentials of each E-mail client, the full path to their E-mail directory in the server, the maximum size allowed for each account and the creation and last accessed date and time.

The *alias* table stores the local aliases or local system users used by Postifx.

The *domain_admins* table stores the administrators for each virtual domain mapped by Postfix.

The *Domain* table stores all the characteristics of each virtual domain in the E-mail server.

The following command is used to create and execute the database:

mysql -uroot -p"password" < postfix-mysql.sql

The command opens the MySQL server with root privileges and loads the SQL script file to create the database.

The MySQL script file is shown in Appendix A.6.

4.1.6. Bind9

Bind9 [45] is one of the most used DNS servers across the Internet providing a robust an stable architecture. It also provides the implementation of all the major components of the Domain Name System like the DNS resolver library and tools for verifying the correct operation of the server.

The Bind9 DNS server can be installed through the Linux repositories using the "apt-get install" command in terminal or, as in the case of this work, using the "Synaptics" tool (graphical version of the apt-get install command) found in Ubuntu.

The main operations that were done to configure the server are as followed:

- Edit the *named.conf.local* file found in the "/etc/bind/" directory to add the DNS Primary Master Zones. This file includes the information of the name of the DNS Zone and the full path for the file with the main characteristics of the zone.
- Configure the DNS zone file and the reverse zone file. These files are usually found in the same directory as the *named.conf.local* file (for this work these files are stored in "/etc/bind/zones/"). These files have information regarding the zones configured for the DNS server and allow the process of DNS Lookup and Reverse DNS Lookup to run.

The process of configuration of the DNS server is shown in the Appendix A.7.

4.1.6.1. Domain Name System (DNS)

DNS [46, 47] is a distributed database that gives the possibility of "translating" host names (names of Internet sites, for example) into IP addresses. The DNS protocol also gives the possibility of doing the reverse operation, translating IP addresses into host names.

DNS organizes the host names into domains with a hierarchical structure. Figure 4.2 is an example of the structure used by the DNS protocol.

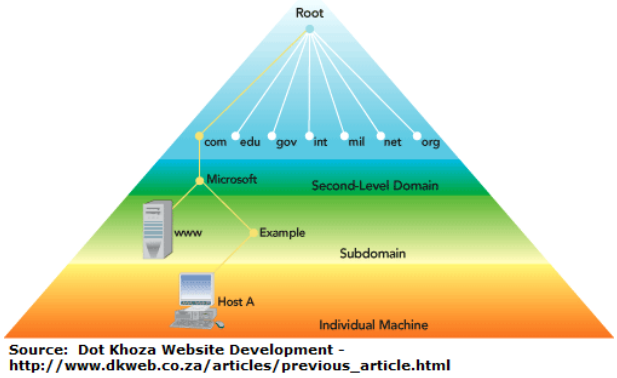


Fig. 4.3 - DNS structure

When a client enters a URL, the request that is made goes through a local name server so that the requested URL is translated into an IP address. If the local name server cannot translate the requested URL, it sends a request for name resolution to the Internet. The response of this request is then kept by the local name server in cache in order to be used for later requests.

4.1.7. Net-SNMP

The Net-SNMP tool [48] was developed at the Carnegie-Mellon University and is a set of applications to implement SNMPv1, SNMPv2 and SNMPv3 using IPv4 and IPV6 networks. This tool allows the exchange of information between the manager and the agents inside the network.

The requests made by the manager can be made by introducing the commands needed inside a terminal. For this work, in order to do this process in an automatic and efficient way, two "shell script" files were created in the server to collect the SNMP data from all the supported network devices.

The first file, "control.sh" controls the instants of the SNMP requests. Every 10 seconds, the second file, "snmp_b.sh", is executed to collect the SNMP data using the "snmpwalk" command.

The information gathered from the routers can be seen in the following table:

Iso.org.internet.mgmt. mib-2.(x) name	**OID**	**Fields**	**Description**
ifXEntry	.1.3.6.1.2.1.31.1.1.1	ifInOctects	Information of network traffic in
		ifInUcastPkts	

		ifInNUcastPkts	every interface of the device.
		ifOutOctects	
		ifOutUcastPkts	
		ifOutNUcastPkts	
		ifOutDiscards	
		ifOutErrors	

Table 4.1 - data collected from the routers via SNMP

An example of the script files "control.sh" and "snmp_b.sh" can be viewed in the Appendix A.8.

4.1.7.1. Simple Network Manager Protocol (SNMP)

SNMP [49] is a network management protocol that allows network devices under the IP protocol to be remotely managed. This protocol consists in the interaction between three components:

- Network management System (NMS): An application that monitors and controls the managed devices gathering the necessary data. For this work the NMS will be installed on the server.
- SNMP Agent: It is a software module that exists in a managed device that compiles the information from the device where it resides. This agent also translates the information stored in the MIB to a language that the SNMP can understand.
- Managed Devices (MD): Network equipment that contains an SNMP Agent.

The following figure shows a typical architecture for network management [50].

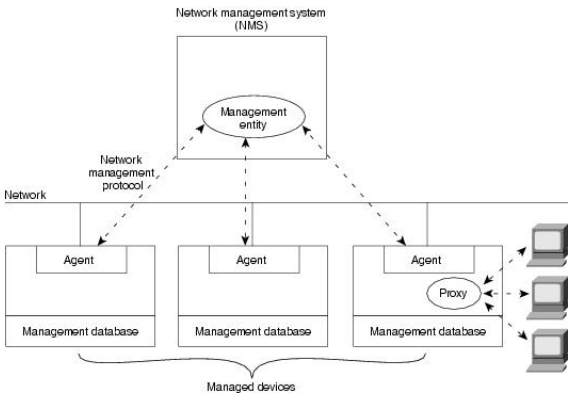


Fig. 4.4 - Network management architecture

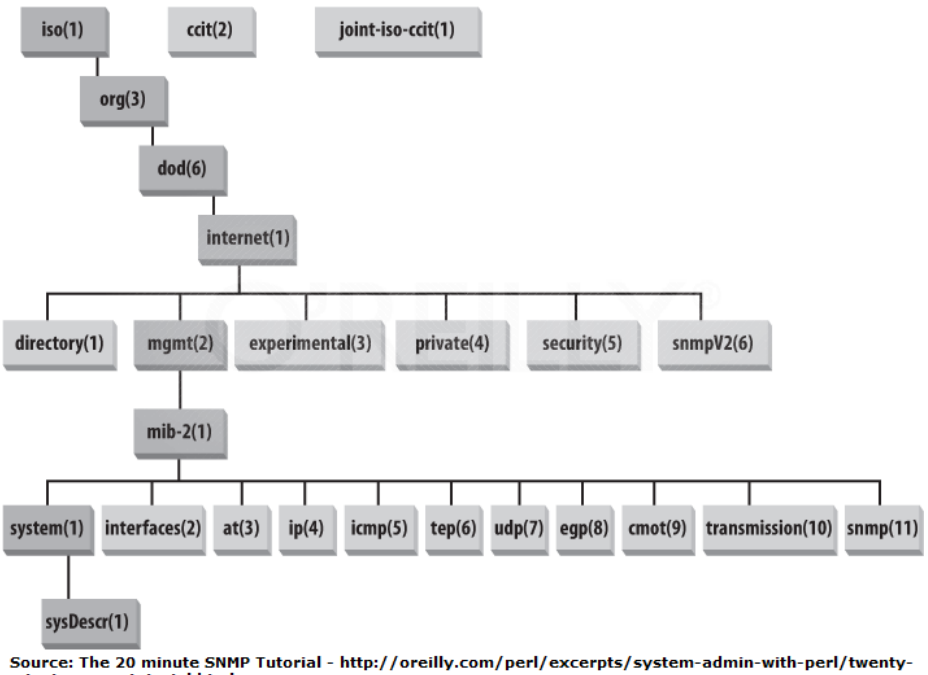
The connection between the NMS and the Agent is made through an UDP link. The choice of using UDP is based on the fact that it has less overhead so that the network is not overloaded by the management system. The ports used by the protocol are:

- Port 161: used for sending the request from the NMS to the Agent and receiving the response from the Agent to the NMS.
- Port 162: used for receiving SNMP traps.

4.1.7.2. Management Information Base (MIB)

The set of SNMP objects is known as MIB [51]. The MIB has a tree structure, where general information is located at the top and the details are found on the ramifications of the tree. The objects definition is independent from the communication protocol that is used, allowing the creation of new sets of MIB variables, defined as "standard" for new devices or new protocols. Each device fabricant can implement their own MIB in order to define objects that need to be managed inside a network.

An example of the MIB tree is shown in the following figure:



Source: The 20 minute SNMP Tutorial - http://oreilly.com/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html

Fig. 4.5 - MIB tree

The supported MIB list for the network devices used in this work can be found in Appendix G.

4.2. Configuration of the Network Devices

4.2.1. Interface Configuration

The following list shows the commands used to configure the routers interfaces and a brief description of each one.

Command	**Description**
Interface *name*	Allows configuring the interface of the router given by the *name* field.
ip address *address netmask*	Associates the IP address and netmask given by the *address* and *netmask* fields.
no shutdown	Sets the interface to be always on.

Table 4.2 - Interface configuration commands

4.2.2. DHCP Server Configuration

For this work, the routers inside the emulated environment were configured as DHCP servers of their LANs so that the end-hosts can obtain their IP addresses via DHCP. To do this, the following commands were used:

Command	**Description**
service dhcp	Enters the DHCP configuration mode.
ip dhcp pool *name*	Associates a name to the DHCP pool to use. This name can be also the used subnet.
network *subnet*	Defines the range of IP address that the router can dynamically allocate using the subnet given in the *subnet* field.
dns-server *address*	Configures the DNS server IP address

	given by the *address* field.
lease 1	Configures the lease time of the IP addresses that are given through DHCP. The default is one day.
exit	Exits the DHCP configuration mode.
ip dhcp excluded-address *addr1 addr2*	Configures a range of IP address that cannot be used in the DHCP pool (the IP address of the routers' interfaces, for example.

Table 4.3 - DHCP configuration commands

4.2.2.1. Dynamic Host Configuration Protocol (DHCP)

DHCP [52] is a protocol used for the dynamic attribution of IP addresses to clients. Besides renting the IP address to the clients, the DHCP can also configure the netmask, default gateway, DNS servers and DNS domains.

The process of "IP lease" in DHCP can be divided into four phases:

- DHCP Discover: This message is encapsulated in a BootP request packet and is used to discover the DHCP servers in the network. In this phase, the client can also indicate a preferential IP address.
- DHCP Offer: the message is encapsulated in a BootP reply packet. The server indicates an IP address that can be leased.
- DHCP Request: Also encapsulated in a BootP request packet. After choosing from the different DHCO offers received, the client indicates the intended IP address.
- DHCP Acknowledge: Encapsulated in a BootP reply packet. The server indentifies the lease of the requested IP address and also provides the client with a set of other information needed for configuration.

4.2.3. OSPF Configuration

All the routers involved in the emulation (real and emulated) were configured with the OSPF protocol to construct their routing tables and the topology map of the network.

The following table shows the commands used to configure OSPF.

Command	Description
router ospf *number*	Defines the OSPF process and assigns a process number given by the *number* argument.
router-id *id*	Assigns the router id number given by the *id* argument (IP address format). When the router-id is not configured, the default router id is the highest IP address on the router interfaces.
network *subnet wildcard* area *number*	Tells the OSPF process the networks that can be directly accessed by the router interfaces and defines the area of the OSPF process.

Table 4.4 - OSPF configuration commands

4.2.3.1. Open Shortest Path First (OSPF)

The OSPF [53] is a link state protocol used to create a database in each router of the network topology and uses the Dijkstra's algorithm[2] to calculate the minimum cost paths. The necessary information for the construction of the database is gathered using a flooding process. The main advantages of the OSPF, when compared to other routing protocols, are the OSPF quickest convergence and the fact that it guarantees loop-free routing.

The topological information is sent through Link State Advertisements (LSAs) contained in the Link State Update (LSU) packets. There are five types of packets used by the OSPF protocol:

- Hello: establishes the neighborhood relationships between the neighbors.
- Database Description: sends the Link State summary.
- Link State Request: asks for the content of a Link State input.
- Link State Update: sends the content of a Link State input.
- Link State Acknowledgement: confirms the reception of a Link State Input.

[2] The Dijkstra's algorithm was developed by Edsger Dijkstra. This algorithm if used in routing because it solves the single-source shortest path problem producing a shortest path tree.

4.2.4. SNMP Configuration

In order to gather the MIB information about all the network devices' interfaces present in the emulation, the SNMP Agents are enabled in each device. To enable them, the following set of commands is used after writing the desired configuration in the devices.

Command	**Description**
snmp-server community *string* RO	Enables the SNMP Agent in Read-Only mode (RO) with the community string given by the *string* argument.
snmp ifindex persist	Enables the SNMP ifIndex persistence.

All the necessary configurations of the routers can be written into a single text file and the copied to each router console via terminal. An example of these configuration files is shown in Appendix B.1.

4.3. Configuration of the PCs hosting the emulated network

In the first tests that were made to the emulation scenarios it was noticed that the only responses received from the server were the responses from the requests of the first end-host. The explanation for this behaviour lies on the fact that the Linux routing table establishes priorities for the configured routes. All the end-hosts were, in fact, sending their requests but they were all arriving to the first end-host and the end-host dropped this information because it did not requested it.

In order to solve this problem, the routes for each end-host need a set of rules for incoming and outgoing traffic, thus forming a routing table for each one of them. When packets arrive, the system will look for the matching set of rules in order to properly deliver the packets to the end-hosts. Also, the arp_filter file for each interface (the interfaces for the end-hosts and the bridge), found in the "/proc/sys/net/ipv4/conf/interface_name/" directory, needs to be enabled (set to "1" because the default is "0") so that all the interfaces ARP requests can be correctly answered.

Finally, let us mention an important remark about the arp_filter file. When obtaining the IP address of the interface through DHCP, the arp_filter file has to be set to "1" after the interface receives its IP address because the DHCP client (in the case of this work, the *dhclient* command) will reset the value of the file to "0".

The following set of commands shows the process of the routing table creation and the arp_filter activation:

```
route add default gw 192.168.5.1 dev tap1

ip route add 192.168.5.0/24 dev tap1 table 2
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap1 table 2
ip rule add from 192.168.5.2 table 2
ip rule add to 192.168.5.2 table 2

echo 1 > /proc/sys/net/ipv4/conf/tap1/arp_filter
```

The *route add* command adds the default gateway for the network device (in this case, *tap1*).

The *ip route add* command is used to add two new entries to the routing table. In the case of the example given above, the command provides information about the tap1's local Class-C subnet and the default gateway for the interface.

The *ip rule* command adds the rules for the incoming and outgoing network traffic of the interface.

Finally, the *echo* command sets to "1" the arp_filter file of the interface.

An example of the configuration for all the interfaces in a scenario is shown in Appendix C.2.

4.4. Emulation Scenarios

All the network scenarios used for this work are based on the real network topology of the University of Aveiro infrastructure; they are simplified models of the real topology, because for this work will only test simple IPv4 wired topologies in order to prove that an emulation platform can be used to obtain the Ground Truth corresponding to a network scenario.

4.4.1. Scenario A

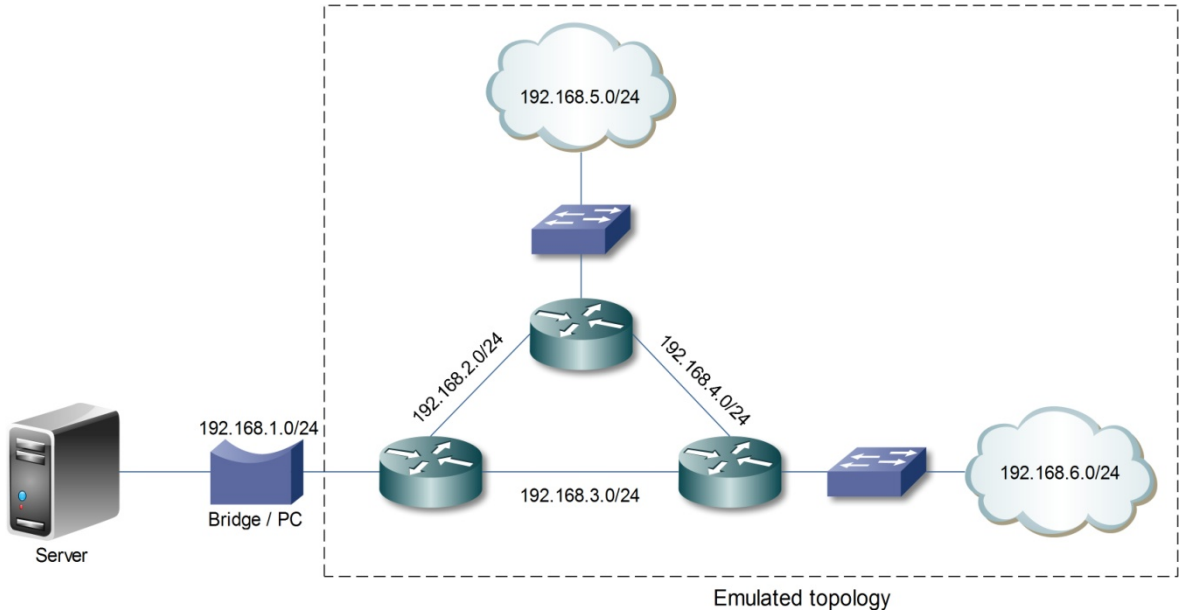


Fig. 4.6 - Network Topology for Scenario A

This scenario represents the interaction between the two main routers of two departments of University of Aveiro and a backbone router connected to the server. The departments can interact between them and if one of the links between the three routers goes down, that is a backup route passing through the link of the other department, so that access to the server is guaranteed.

The clouds represent the LANs inside each department. For this scenario, the LANs have a total of five end-hosts, each one making random requests to the server, and an end-host that can be connected in any part of the network. This movable end-host is only used for performing security attacks to the network.

By using an end-host exclusively for triggering network attacks it is possible to easily extract from the captures all the packets generated by the attack in order to analyze them separately from the rest of the network traffic that was generated during the emulation.

For this scenario, the end-hosts can send DHCP requests to obtain their IP address and can also send HTTP and FTP requests randomly to the server. The emulated routers belong to the 3745 series (c3745-advipservicesk9-m.124-4.T1.bin). The services that were configured in the routers and the server can be seen in the following table:

Routers	Server
DHCP server	APACHE
OSPF	BIND9
SNMP Agent	ProFTPD
	NMS

Table 4.5 - Configured services in the routers and server for scenario A

This scenario was also used to test the PCs hosting the emulated devices in order to look for problems in the emulation and to see if the PCs were able handle all the processing needed for the emulation to run. This scenario generates, approximately, 5GB of network traffic per day. A more detailed version of the network topology can be seen in the Appendix D.1.

4.4.2. Scenario B

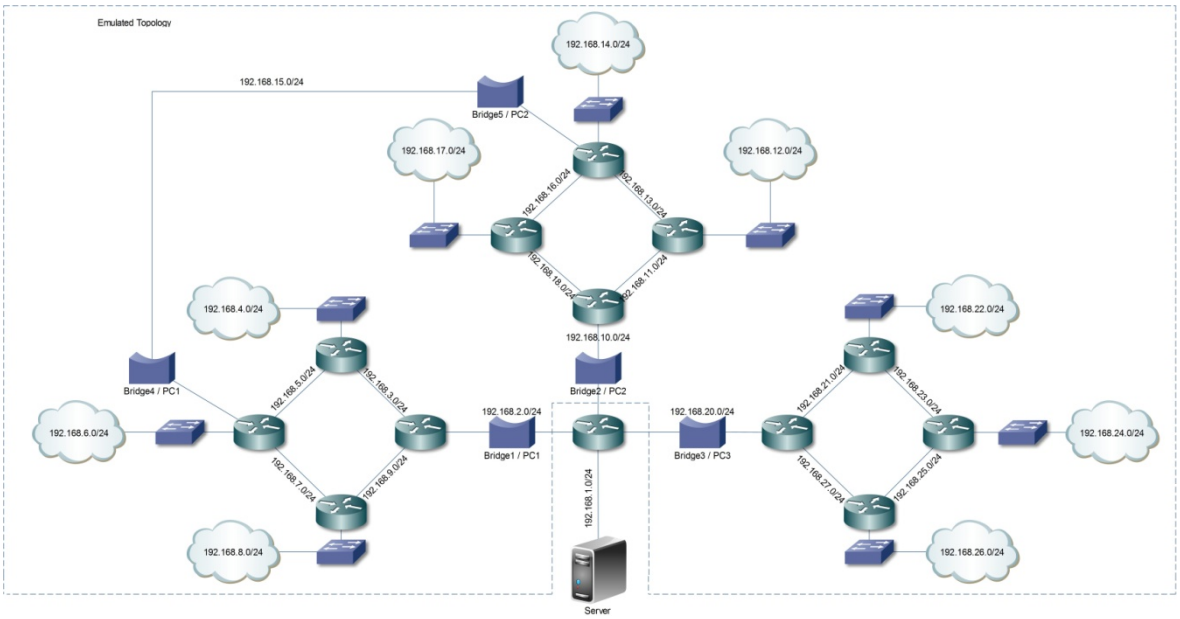


Fig. 4.7 - Network Topology for Scenario B

This topology represents a simplified version of the topology used by University of Aveiro. Each host PC represents a different part of the campus. The backbone network of this topology is formed by 3 emulated routers and a real 3825 Cisco router (IOS version 12.4(3g)). The 3825 Cisco router is directly connected to the server in order to allow communications between each point of the network and the server.

In this topology each part of the campus holds three departments that are connected to the server via different points of the backbone network. Each router representing a department holds a LAN of 6 end-hosts, where 5 of them generate normal network traffic (DHCP, HTTP, FTP, SMTP and POP3 requests) and the sixth end-host connected in each LAN is used for security attacks.

The services that were configured in the routers and servers can be seen in the following table:

Routers	**Servers**
DHCP Server	APACHE
OSPF	BIND9
SNMP Agent	ProFTPD
	NMS
	POSTFIX

Table 4.6 - Configured services in the routers and server for scenario B

This scenario can generate 40GB of network traffic per day. A detailed version of the network topology can be seen in the Appendix D.2.

4.4.3. Scenario C

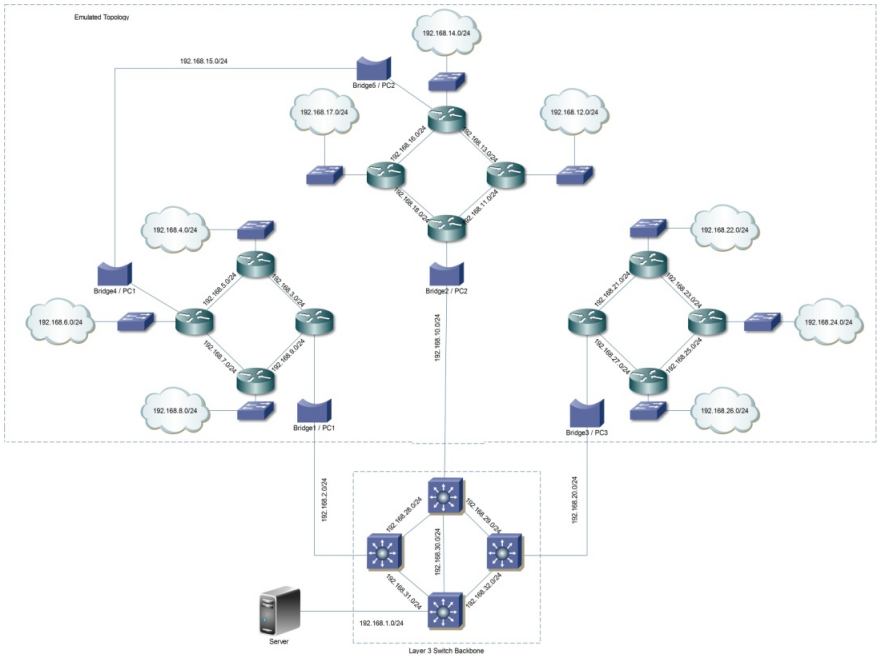


Fig. 4.8 - Network Topology for Scenario C

This scenario is very similar to the previously presented one and also represents a simplified version of the network topology used by University of Aveiro: the difference lies in the fact that the backbone network for this scenario is composed by four real catalyst 3750 Metro series Layer 3 (IOS version 12.1(14r)) switches and 3 emulated 3745 Cisco routers.

The Layer 3 switches are high-performance devices used for network routing and have very little differences from typical routers. These switches can support the same routing protocols as normal routers and have the advantage of enabling the usage of InterVLAN routing in a single equipment, besides costing less than the sum of a normal router and a normal switch.

A detailed version of the network topology can be seen in the Appendix D.3.

4.5. Network Traffic Scenarios

The topologies were tested using a series of different network traffic scenarios, each having a 24 hours duration. These Scenarios can be divided into three main groups:

- **Clean Scenario:** This scenario generates network traffic without anomalies.
- **Port-Scan Attack Scenario:** The network traffic generated in this scenario is a mixture of normal network traffic with Port-Scan security attacks made from a single or multiple end-hosts to the server.
- **Snapshot Attack Scenario:** The network traffic generated in this scenario is a mixture of normal network traffic with Snapshot security attacks made from a single end-host to the server.

The tool used in this work to make the port scan attack is called *Nmap. Nmap* (Network Mapper) is an open source tool used for network exploration and security auditing. In order to do this task, Nmap uses raw IP packets to determine a series of characteristics of the scanned host, like the services offered by the host, the OS that is running in it or the type of firewall used.

The *Nmap* also retrieves a list with the port numbers used by the host, the service running on those ports and their states. The states can be classified as open, filtered, closed or unfiltered.

The open state means that an application in the target is listening for connections on that port, while filtered means that the port is being blocked by a firewall. As the name indicates, the close state is used when the port is closed and the ports are classified as unfiltered when the ports respond to the *Nmap's* probes. The port scans are normally used by hackers to discover open ports in the target in order to gain access to the target's end-host or they can also be used to create DDoS attacks.

The syntax of the Nmap command used for this work is shown below:

nmap -S 192.168.5.7 -T -sV -v -O www.simulation.net

In the given example the nmap is going to do port scan to the domain *www.simulation.net* using the IP *192.168.5.7* as the source address (*-S* option). The *-T* parameter sets the timing template that is going to be used. The *-sV* parameter enables the version detection mode. This is used to identify services on the target. The *-v* parameter enables the verbose mode allowing *Nmap* to print more information about the scan in progress. The *-O* parameter enables the OS detection in *Nmap.*

The **Port-Scan Attack Scenario** is divided into three types of port scans depending on the timing template that is used:

- **Sneaky (T1):** used for IDS evasion. A complete port scan with this timing template can take up to five hours to end because the scans are not made in parallel mode and the waiting time between each probe that is sent to the server is 15 seconds.
- **Normal (T3):** this is the default timing template used by *Nmap.* It allows port scan parallelization process.
- **Aggressive (T4):** as the name suggests, it is an aggressive port scan where the scan delay of the probes does not exceed 10 milliseconds (for TCP ports).

For each type of Port Scan Attack that is used in this work there is also a division of different types of attacks. This division and a brief description of each sub-type are given in the following table:

Type of attack	**Description**
1-to-1	A single attacker making a port scan to a single target.
1-to-N	A single attacker making port scans to

	multiple targets. Normally this is used against someone with more than 1 associated IP address.
N-to-1	Multiple attackers against a single target.
N-to-N	Multiple attackers against a range of targets.

Table 4.7 - Attacks made for each time division

The **Snapshot Attack Scenario**, as the name suggests, consists of making snapshots of the target monitor contents. These snapshots are then uploaded to a destination of the attacker preference and can be used to discover confidential information about the target. Some "worms" that can be found on the Internet act like this: while consuming the target resources, they can "catch up" snapshots of the target.

In this scenario, the snapshot attack is divided into two categories:

- **Exponential:** In this attack each time the users does a "click" in some part of the monitor, a snapshot of a small area around the pointer is taken (this name derives from the fact that intervals between user clicks are exponentially distributed in order to model human behavior in the best possible way.).
- **Periodical:** in this attack, a snapshot of the entire screen is taken in periodical time intervals.

Because these attacks are uploads of relatively small images, in this we emulate this attack by using a virtual end-host inside the emulation that uploads image files to the server using an exponentially distributed time interval, in the first case and a periodic time interval in the second case.

The tool that was used to emulate this behavior (and also for the FTP download and upload traffic that will be explained in the following sub-section) is called *cURL.*

cURL is a command line tool for transferring files with a URL syntax. It currently supports FTP, HTTP, FTPS, HTTPS, TELNET and many others protocols that uses the URL syntax.

The cURL syntax for uploading a file through FTP is shown below:

curl --interface <interface> -T <file> <URL_FTP> --user <user:passwd>

The *--interface* parameter allows performing the FTP upload using a specific interface.

The *-T* parameter transfers the local file to the remote FTP URL.

The *--user* parameter specifies the user and password to use for server authentication.

For this work, the following attack script files were developed:

- nmap_aggressive.sh.
- nmap_normal.sh.
- nmap_sneaky.sh.
- snap_exp.sh.
- snap_per.sh.

The contents of these script files are shown in the Appendix E.1.

4.6. Procedure

After creating all the necessary virtual end-hosts and bridges, configuring the default route in the server and connecting all emulated and real network devices the following procedure must be followed to start the emulation:

- Start the emulation inside GNS3 and configure all the network devices (emulated and real).
- Run the DHCP script file so that all the end-hosts can obtain their IP addresses.
- Configure the different routing tables and rules for each virtual end-host.
- In the server, start *tshark* in order to capture all the packets and execute the SNMP script file for gathering the MIB data.
- In the host PC, start *tshark* to capture all the packets in all the virtual interfaces.
- Execute the *start_script.sh* file so that the HTTP, FTP, SMTP and POP3 traffic generators script files are executed.
- In the case of the attack scenarios, execute the script file corresponding to the attack that is going to be launched.

The scripts files involved in the traffic generation are the following:

Dhcp_script.sh: this file uses the *dhclient* tool to obtain the IP addresses for all the virtual interfaces. The syntax for this command is the following:

dhclient interface_name

Start_script.sh: the script responsible for starting the traffic generator scripts. The script executes the exponential distribution tool[3] with the help of the *gnome-terminal* command.

The syntax used in this work for the command is show below:

gnome-terminal --working-directory=$path -x $file_path

The *gnome-terminal* opens a Linux terminal in the directory specified by the *--working-directory* option. The *-x* option allows the execution of a file.

This script controls the execution of all the traffic generator scripts of all the end-hosts involved in the emulation. The *sleep* command is used in this script so that the different executions of the exponential distribution tool can begin at different times in order to generate different exponential distributions for each traffic generator script.

get.sh: One of the traffic generator scripts. When used for generating HTTP traffic the used command is the following:

wget --bind-address=addr URL

The *wget* tool is a network downloader and supports the HTTP, HTTPS and FTP protocols. The *--bind-address* parameter allows the command to perform the requests using a specific IP address.

When the *get.sh* script is used for FTP download, the command used is *cURL* and the syntax for downloading via FTP is the following:

Curl --interface <interface> <FTP_URL> --user <user:passwd> -o <file>

The syntax is very similar to the upload case shown in sub-section 4.5. The *-o* option indicates the output file for the command.

[3] The exponential distribution tool is a C-based program developed by Eduardo Rocha. The program creates an exponential distribution based on the system clock and the exponential mean set by the user. Each point of the exponential distribution is an event represented in seconds. Each time an event occurs the program executes a shell script that has to be in the same directory as the program.

For this work, it was assumed that a typical user makes a new HTTP or FTP request, in average, in time intervals of 120 seconds [5]. This value will be used by the emulation tool as the mean to create the exponential distribution for the HTTP and FTP requests.

make_mail.sh: this script is responsible for generating the outgoing e-mail for each one of the end-hosts. This script depends on three files to work properly: the *mailnum*, the *bigmail.txt* and the *smallmail.txt*. The *mailnum* file stores the last e-mail sent so that the script knows which e-mail to send next. This is done by using an "if, then, else" statement to compare the number inside the file with the number "2". If the number is less than "2" then, the *smallmail.txt* file will be sent, if not, the *bigmail.txt* file is sent.

It was assumed that an average user sends an E-mail, in average, every 10 minutes [3]. This value is used (in seconds) to create the exponential distribution to generate SMTP requests.

see_mail.sh: this is the script file responsible for reading the e-mail. This script depends on two files. The *mailfile* is where the server responses and downloaded e-mails are stored and the *numfile* stores the index numbers of the e-mails in order to be consulted. The echo command is used to insert the user and password for the e-mail account and to obtain the list of e-mails in the account. After downloading the e-mails with the *retr* command, the messages are deleted using the *dele* command.

The time interval used in this work to check for new E-mail messages is 10 minutes. This value is the default time used by almost all the E-mails clients like Thunderbird or Microsoft Outlook.

The contents of these script files are shown in Appendix E.1.

4.7. Summary

This chapter has shown all the configurations in the server, routers and host PCs that are necessary for the emulation. Besides, it has also shown the network topologies, the network traffic scenarios and the script files that were used for this work.

5. Collected Data

The main purpose of this work, as previously explained, is the creation of a platform in a controlled environment where topologies, protocols, anomalies and network behaviour can be analyzed to gather a trustworthy data set that can be considered the network Ground Truth. The following sub-sections will show examples of the collected data that can be gathered from the virtual interfaces in order to characterize the behaviour of a single user inside the scenarios that were emulated in this work and examples of the collected data from the server (captures, logs and SNMP information).

The capture files gathered from the client PCs and the server can be used to make a statistical analysis of the traffic that is generated by a single user and the aggregated traffic at the server.

In order to do this, a one hour statistical analysis of the traffic generated was made with the help of the *mergecap*, *tshark* and *Matlab* tools.

The *mergecap* tool was used to merge the 30 minutes capture files into a 1 hour capture file. The syntax of this tool is shown below.

```
mergecap -w merged_file.pcap first_file.pcap second_file.pcap
```

The *-w* option sets the output filename. This is a mandatory option for the tool to work.

The *tshark* tool is a text-based capture tool that is also used to gather statistics (in bytes and frames) of the capture files that were previously merged. For this part of the work, the *tshark* command can have two different syntaxes:

```
tshark -r cap_file -q -z io,stat,interval,[filter] > output_file.txt
tshark -r cap_file -q -z "io,stat,interval,[filter]&&[filter] > output_file.txt
```

The *-r* option tells *tshark* to read from the capture file (cap_file). The *-q* option is used to tell *tshark* to print only the statistics that are collected by the *-z io, stat, interval*. This last option is used to gather packet/bytes statistics in a specific sampling time (for this work, the sampling time used was 100 milliseconds). This time is given by the *interval* parameter and can have a millisecond resolution.

The first previously shown syntax is used when the *tshark* command is used with no filters or only with one filter. The second syntax is used when the statistics depends on more than one filter.

The output of this command is stored in a text file that will be used by *Matlab* to create graphics for the collected statistics. An example of the output file can be seen in figure 5.1.

```
================================================
====
IO Statistics
Interval: 0.100 secs
Column #0: http&&ip.addr==192.168.5.2
                  |   Column #0
Time              |frames|  bytes
000.000-000.100         0          0
000.100-000.200         0          0
000.200-000.300         0          0
000.300-000.400         0          0
000.400-000.500         0          0
000.500-000.600         0          0
000.600-000.700         0          0
000.700-000.800         0          0
000.800-000.900         0          0
000.900-001.000         0          0
001.000-001.100         0          0
001.100-001.200         0          0
001.200-001.300         0          0
001.300-001.400         0          0
001.400-001.500         0          0
001.500-001.600         0          0
001.600-001.700         0          0
```

Fig. 5.1- Example of the statistics output file

The gathered statistics are showed in columns and the header of the output file shows the filter that was used to create the statistical analysis.

Matlab is used to generate graphics for the gathered statistics. For this work, this will be done by loading the text file into *Matlab* using the *load* command and calling an M-file called "read_data.m". The file contains *Matlab* commands to create the intended graphics. The syntax is show below.

Read_data(loaded_file, number_of_column)

This function has two parameters: the first parameter is a variable that represents the loaded file and is stored in an Nx3 matrix; the second parameter represents the column that is going to be used, together with the sampling time, to generate the graphic. Using the number "2" will generate a graphic based on the number of packets and using the number "3" will generate a graphic based on the number of bytes. The M-file can be seen in Appendix F.

5.1. Collected data from a virtual interface

As previously explained, the virtual interfaces used in this work have two purposes: the interconnection between real equipment and the emulated topologies and the emulation of end-hosts connected to the network. In real networks, a user can ask randomly for any type of request to the server, for example, a user can see a webpage while waiting for the e-mail client to verify his e-mail accounts. The end-hosts used for this work can emulate this behaviour with the help of the traffic generator program explained in the previous chapter.

So, for the first scenario, the end-hosts can obtain their addresses via DHCP, visualize websites and can upload and download files to/from a FTP server and for the second scenario, the end-host can also send and receive e mails.

To store this data, the *tshark* tool was used. The syntax of the command used for the captures is shown below:

```
tshark -b duration:1800 -i any -s 100 -w $path_output
```

When *tshark* is used with this syntax it allows to do a multiple file capture (*-b* parameter), where the capture stops and starts in the next file when a limit is reached (in this case, each capture file has a 30 minute length). The *-i* option allow to choose the capture interface , the *-s* option sets the default snapshot length when capturing live data and the *-w* option allows to write the raw data to the output file.

The characterization of the requests made by a single interface can be seen in the following sub-sections.

5.1.1. HTTP Profile

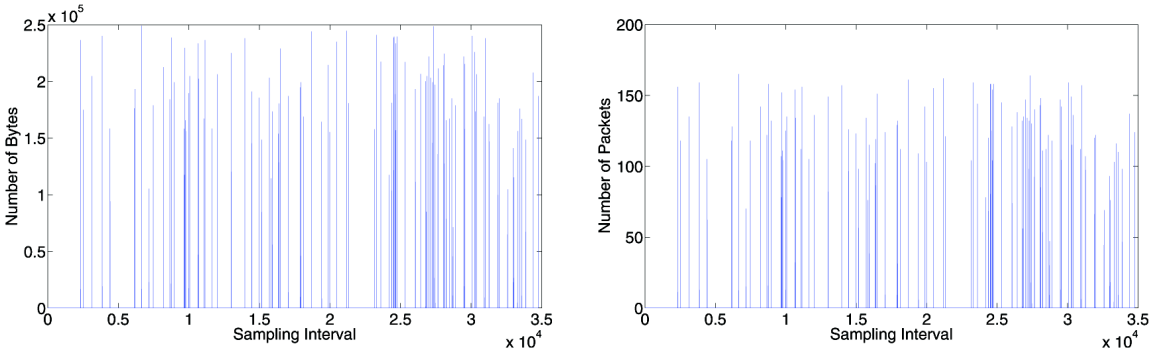


Fig. 5.2 - HTTP profile of a single interface in number of bytes and number of packets

The above graphics represent the HTTP profile of a single user over an hour and are from a bidirectional capture (both the requests from the user and the replies from the server are shown in the graphics). The requests from the user are random requests (in terms of time instants) and, as referred in the previous chapter, they are exponentially distributed with a mean value equal to 120 seconds.

The first peak represents the HTTP request made by the user to the server (upload traffic) and the subsequent burst of packets that appears immediately after this peak represents the response of the server (download of the website).

5.1.2. FTP Profile

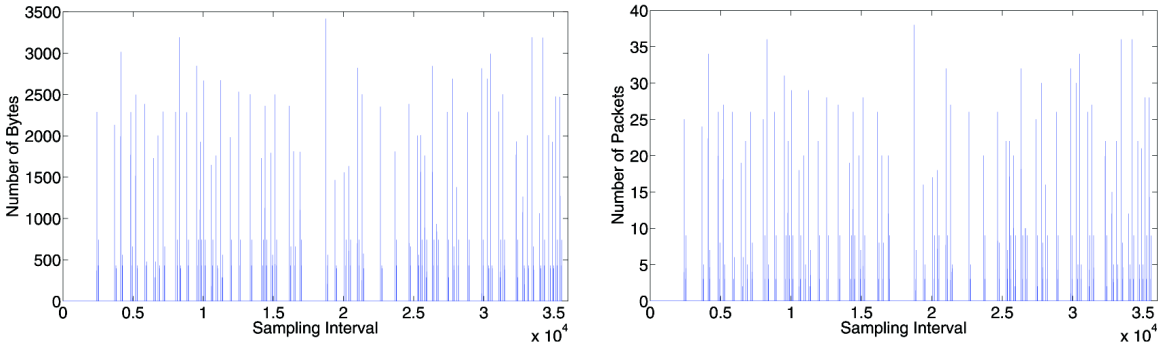


Fig. 5.3- FTP profile of a single interface in number of bytes and number of packets

The above graphics represent the FTP profile of a user that is downloading small files from a FTP server. As explained in the previous chapter, because the scenarios used for the emulation are simplified models of the network topologies inside University of Aveiro, the FTP download and upload exemplified in this profile represent

a single student accessing the ARCA site inside the University, that constitutes a FTP site used by students to store small files.

5.1.3. DHCP Profile

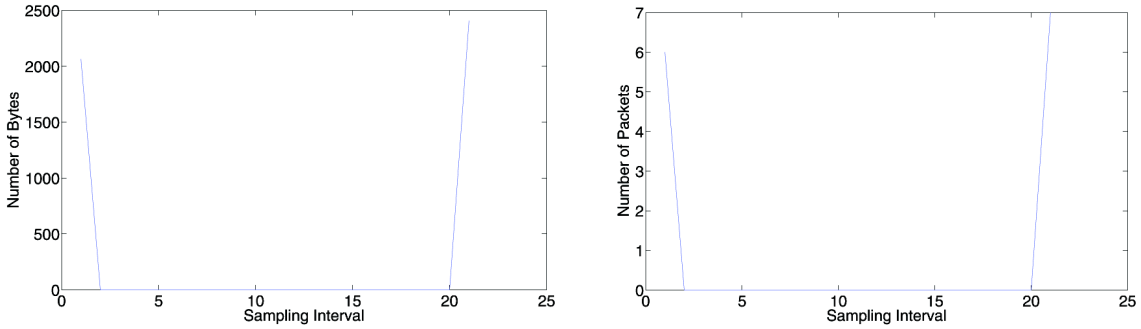


Fig. 5.4 - DHCP profile of a single interface in number of bytes and number of packets

The above graphics represent the number of bytes and packets when a single user asks for the IP address through DHCP request. The first peak represents the client DHCP request sent to the broadcast address; the DHCP server catches the message and sends the DHCP reply to the client with the attributed IP address, here represented by the second peak. The reason for only seeing two peaks in these graphics is because the DHCP renewal time used by default in the *dhclient* command and in the Cisco routers is 1 day.

5.1.4. DNS Profile

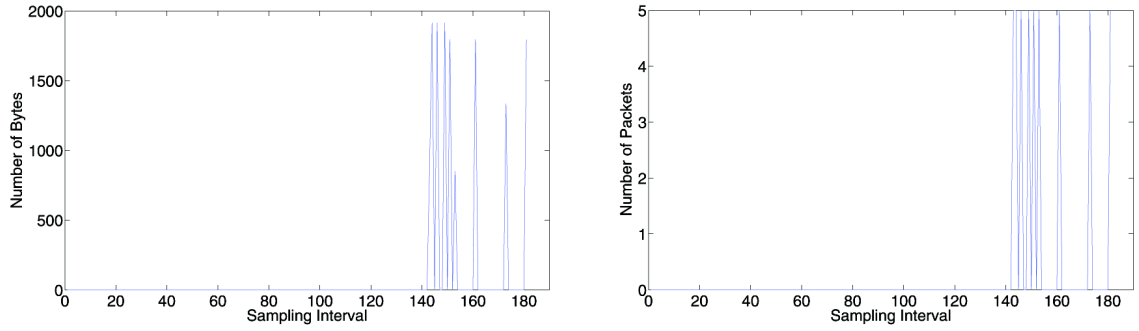


Fig. 5.5 - DNS profile of a single interface in number of bytes and number of packets

The above graphics represent the number of bytes and packets when a single user asks for a URL for the first time. When the IP address is not in the DNS cache, the request goes to the DNS server in order to associate the URL of the site to its IP address and this information comes back to the end-host where it is stored in its DNS cache.

5.1.5. E-Mail profile

The e-mail profile of a single user is divided in two parts: the SMTP profile for sending e-mails and the POP3 profile for the reception of e-mails.

5.1.5.1. SMTP Profile

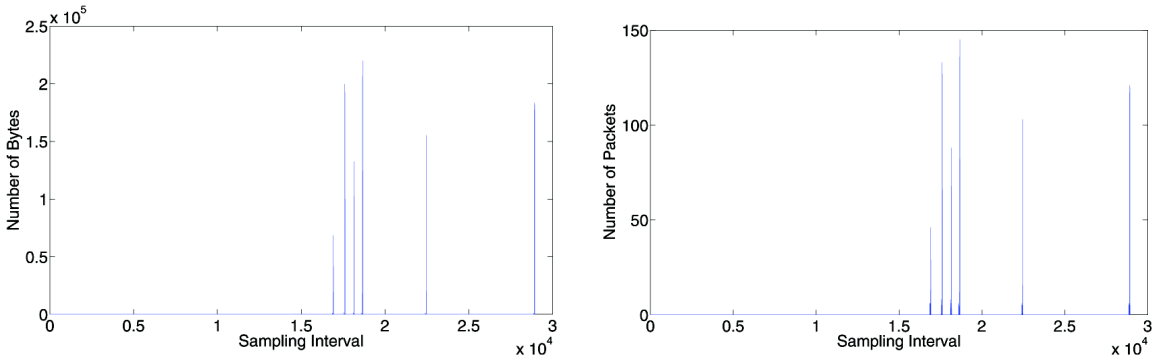


Fig. 5.6 - SMTP profile of a single interface in number of bytes and number of packets

As explained before, the SMTP protocol is used for sending e-mails. The first peaks of the graphics correspond to the user access to the server and the larger peak corresponds to the e-mail that was sent (upload to the server).

5.1.5.2. POP3 Profile

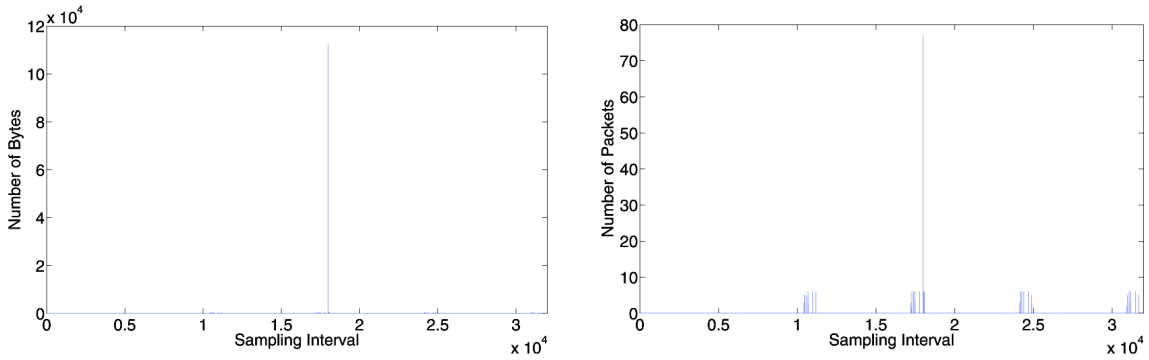


Fig. 5.7 - POP3 profile of a single interface in number of bytes and number of packets

The POP3 protocol is used for the reception of e-mails. As explained in the previous chapter, the time interval used for the verification of e-mails is a standard interval that is used in the majority of e-mail clients, like Microsoft Outlook or Mozilla Thunderbird.

The peak that is shown in the graphics represents the download of an e-mail to an end-host, the other smaller peaks represent the e-mail client verifying if the server has any new e-mails.

5.1.6. Attack profile (port scan)

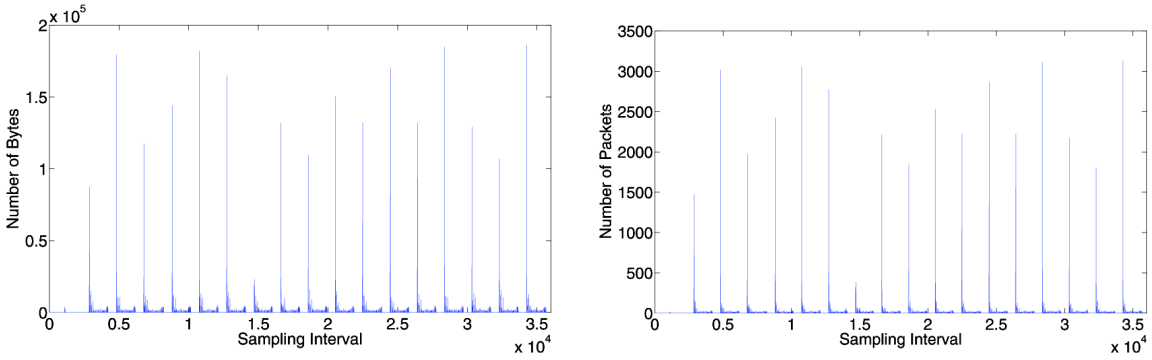


Fig. 5.8 - Port Scan profile of a single interface in number of bytes and number of packets

As explained in the previous chapter, the usage of a separated interface for the attacks simplifies the process of characterizing the attack itself as it can be seen in the previous graphics. The attack shown here is made using the command Nmap in its default behaviour.

Each group of peaks represents a port scan attack made to the server and they are separated by a waiting time of one minute.

5.2. Collected data from the server

5.2.1. Log Files

The Log files can play a vital role in a server and provide vital information on the services running on it, like for example the IP address, date and time of an HTTP request.

The log Files gathered in this work were the log files of the HTTP server (Apache), the DNS server (Bind9), the FTP server (ProFTPD) and the log files of the e-mail server (Postfix).

5.2.1.1. APACHE Log Files

The APACHE log file is called "access_log" and can be found in "/var/log/apache2". This log keeps tracks of all the requests made to the APACHE server.

The format of the access log can be configurable. There are, typically, two types of formats: the common format and the combined format. The parameters for these types of formats are shown in the following table.

Name	Format
Common	"%h %l %u %t \"%r\" %>s %b"
Combined	"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""

Table 5.1 - APACHE log file formats

The "common" format is configured by default. To use the "combined" format, in the APACHE configuration file, the command "CustomLog/access_log common" has to be changed to "CustomLog/access_log combined".

Using the example taken from the APACHE documentation that refers to logs [11] a description of the parameters used in the common log format is shown in table 5.2.

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

Parameter	Meaning	Example
%h	IP address of the client.	127.0.0.1
%l	RFC1413 identity of the client.	-
%u	User id of the person requesting the document. This id is determined by HTTP authentication.	frank
%t	The time when the request was received.	[10/Oct/2000:13:55:36 -0700]
\"%r\"	The request line from the client.	"GET /apache_pb.gif HTTP/1.0"
%>s	Status code sent by the server to the client.	200
%b	Size of the object returned to the client.	2326

Table 5.2 - APACHE Log parameters

An example of the APACHE log file of the server can be seen in the following figure.

```
192.168.4.2 - - [21/Sep/2009:14:33:45 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:48 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:48 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.2 - - [21/Sep/2009:14:33:56 +0100] "GET /CNN.htm HTTP/1.0" 200 103483 "-" "Wget/1.11.4"
192.168.4.5 - - [21/Sep/2009:14:34:04 +0100] "GET /cyberhome.htm HTTP/1.0" 200 25247 "-" "Wget/1.11.4"
```

Fig. 5.9 - APACHE log file

5.2.1.2. ProFTPD Log Files

The ProFTPD has two log files, the TransferLog file called "xferlog" and the SystemLog file called "proftpd.log". Both files can be found in "/var/log/proftpd/".

The parameters of the "Xferlog" file are separated by whitespaces and are described in the following table:

Parameters	Meaning
Current-time	Describes the access date and time in the following format: "DDD MMM dd hh:mm:ss YYYY".
Transfer-time	Duration of the transfer in seconds.
Remote-host	Name of the remote computer.
File-size	The size in bytes of the file being transferred.
Filename	The full path of the file.
Transfer-type	Type of transfer: letter "a" represents ASCII and letter "b" represents binary.
Special-action-flag	Character "C": corrupted file. Character "U": non-corrupted file. Character "_": no specification.
Direction	The direction of the transfer: outgoing (o), incoming (i), deleted (d).
Access-mode	Type of login access: anonymous (a), password required (g), local user (r).
Username	The name of the user.
Service-name	Name of the service (FTP).
Authentication-method	No authentication (0), RFC931 authentication (1).
Authentication-user-id	User id.
Completion-status	Status of the transfer: complete (c), incomplete (i).

Table 5.3 - Xferlog file parameters

An example of the "xferlog" file can be seen in figure 5.10

```
Thu Aug 20 19:40:36 2009 10 192.168.5.2 498112 /home/FTP-shared/download/Screenshot.png b _ o r userftp ftp 0 * i
Thu Aug 20 19:41:50 2009 2 192.168.5.2 278016 /home/FTP-shared/download/Screenshot.png b _ o r userftp ftp 0 * i
Thu Aug 20 19:42:42 2009 1 192.168.5.2 740327 /home/FTP-shared/download/Screenshot.png b _ o r userftp ftp 0 * c
Thu Aug 20 20:22:26 2009 1 192.168.5.2 740327 /home/FTP-shared/download/Screenshot.png b _ o r userftp ftp 0 * c
Thu Aug 20 20:22:35 2009 1 192.168.5.3 740327 /home/FTP-shared/download/Screenshot.png b _ o r userftp ftp 0 * c
```

Fig. 5.10 - Xferlog file

The "proftpd.log" file follows the SystemLog format. These logs are created by the "syslog" service described in RFC3164 [31].

The format in these logs can be divided into three parts: PRI, HEADER and MSG.

The HEADER part has two fields: TIMESTAMP and HOSTNAME. The TIMESTAMP field is the local time and date and it follows the "Mmm dd hh:mm:ss" format. The HOSTNAME field contains de IP address (IPv4 or IPv6) or the hostname.

The MSG part also has two fields: the TAG and the CONTENT field. The TAG field contains the name of the service or program that generates the messages and the CONTENT field shows the details of those messages.

An example of the "proftpd.log" file can be seen in the following figure.

```
Nov 15 19:34:28 sim-server.simulation.net proftpd[24153] sim-server.simulation.net (::ffff:192.168.1.2[::ffff:192.168.1.2]): USER simulation: Login
successful.
Nov 15 19:34:28 sim-server.simulation.net proftpd[24153] sim-server.simulation.net (::ffff:192.168.1.2[::ffff:192.168.1.2]): Preparing to chroot to
directory '/home/FTP-shared'
Nov 15 19:34:41 sim-server.simulation.net proftpd[24153] sim-server.simulation.net (::ffff:192.168.1.2[::ffff:192.168.1.2]): FTP session closed.
Nov 15 20:00:51 sim-server.simulation.net proftpd[24544] sim-server.simulation.net (::ffff:192.168.5.3[::ffff:192.168.5.3]): FTP session opened.
Nov 15 20:00:51 sim-server.simulation.net proftpd[24544] sim-server.simulation.net (::ffff:192.168.5.3[::ffff:192.168.5.3]): USER simulation: Login
successful.
Nov 15 20:00:51 sim-server.simulation.net proftpd[24544] sim-server.simulation.net (::ffff:192.168.5.3[::ffff:192.168.5.3]): Preparing to chroot to
directory '/home/FTP-shared'
Nov 15 20:01:02 sim-server.simulation.net proftpd[24544] sim-server.simulation.net (::ffff:192.168.5.3[::ffff:192.168.5.3]): FTP session closed.
Nov 15 20:02:23 sim-server.simulation.net proftpd[24564] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session opened.
Nov 15 20:02:23 sim-server.simulation.net proftpd[24564] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): USER simulation: Login
successful.
Nov 15 20:02:23 sim-server.simulation.net proftpd[24564] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): Preparing to chroot to
directory '/home/FTP-shared'
Nov 15 20:02:34 sim-server.simulation.net proftpd[24564] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session closed.
Nov 15 20:10:56 sim-server.simulation.net proftpd[24775] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session opened.
Nov 15 20:10:56 sim-server.simulation.net proftpd[24775] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): USER simulation: Login
successful.
Nov 15 20:10:56 sim-server.simulation.net proftpd[24775] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): Preparing to chroot to
directory '/home/FTP-shared'
Nov 15 20:11:06 sim-server.simulation.net proftpd[24775] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session closed.
Nov 15 20:11:55 sim-server.simulation.net proftpd[24788] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session opened.
Nov 15 20:11:56 sim-server.simulation.net proftpd[24788] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): USER simulation: Login
successful.
Nov 15 20:11:56 sim-server.simulation.net proftpd[24788] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): Preparing to chroot to
directory '/home/FTP-shared'
Nov 15 20:11:56 sim-server.simulation.net proftpd[24788] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session closed.
Nov 15 20:25:05 sim-server.simulation.net proftpd[2914] sim-server.simulation.net: ProFTPD killed (signal 15)
Nov 15 20:25:05 sim-server.simulation.net proftpd[2914] sim-server.simulation.net: ProFTPD 1.3.1 standalone mode SHUTDOWN
Nov 15 20:25:07 sim-server.simulation.net proftpd[25015] sim-server.simulation.net: ProFTPD 1.3.1 (stable) (built Thu Feb 21 04:50:29 UTC 2008) standalone
mode STARTUP
Nov 15 20:25:34 sim-server.simulation.net proftpd[25030] sim-server.simulation.net (::ffff:192.168.5.4[::ffff:192.168.5.4]): FTP session opened.
```

Fig. 5.11 - proftpd.log file

5.2.1.3. E-Mail Log Files

The E-mail log file is called "mail.log" and it can be found in "/var/log". Like the "proftpd.log", that was previously mentioned, the "mail.log" is generated by the "Syslog" service, described in the RFC3164 [31], and it follows the same format that was previously described.

An example of the "mail.log" file can be seen in figure 5.12.

```
Sep 26 07:57:56 sim-server postfix/smtpd[9506]: connect from unknown[192.168.22.6]
Sep 26 07:57:58 sim-server postfix/smtpd[9506]: D0E581C12E: client=unknown[192.168.22.6]
Sep 26 07:58:01 sim-server postfix/cleanup[11548]: D0E581C12E: message-id=<20090926065758.D0E581C12E@mta.simulation.net>
Sep 26 07:58:03 sim-server postfix/qmgr[7814]: D0E581C12E: from=<conta35@simulation.net>, size=986441, nrcpt=1 (queue active)
Sep 26 07:58:03 sim-server postfix/smtpd[9506]: disconnect from unknown[192.168.22.6]
Sep 26 07:58:03 sim-server postfix/virtual[6466]: D0E581C12E: to=<conta5@simulation.net>, relay=virtual, delay=6, delays=5.9/0/0/0.12, dsn=2.0.0,
status=sent (delivered to maildir)
Sep 26 07:58:03 sim-server postfix/qmgr[7814]: D0E581C12E: removed
Sep 26 07:58:14 sim-server pop3d: LOGOUT, user=conta45@simulation.net, ip=[::ffff:192.168.26.6], port=[46890], top=0, retr=0, rcvd=15, sent=97, time=60
Sep 26 07:58:16 sim-server pop3d: LOGOUT, user=conta13@simulation.net, ip=[::ffff:192.168.8.4], port=[56537], top=0, retr=973846, rcvd=33, sent=986715,
time=70
```

Fig. 5.12 - Mail.log file

5.2.1.4. DNS Log Files

The DNS log file is called "bindQuery.log" and like the previously presented logs, it can also be found in "/var/log/". An example of the DNS log file can be seen in figure 5.13.

```
04-Sep-2009 19:54:00.261 client 192.168.1.3#36208: query: www.simulation.net IN A +
04-Sep-2009 19:54:00.750 client 127.0.0.1#49449: query: 8.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:01.329 client 127.0.0.1#39857: query: 6.6.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:01.479 client 127.0.0.1#34016: query: 7.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:01.506 client 127.0.0.1#33561: query: 8.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:03.996 client 127.0.0.1#42574: query: 7.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:04.629 client 127.0.0.1#34015: query: 7.6.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:04.631 client 127.0.0.1#54819: query: 7.6.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:04.804 client 192.168.1.3#42125: query: www.simulation.net IN A +
04-Sep-2009 19:54:05.598 client 192.168.1.3#43469: query: www.simulation.net IN A +
04-Sep-2009 19:54:05.615 client 127.0.0.1#46501: query: 5.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:05.787 client 127.0.0.1#49690: query: 8.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:05.817 client 192.168.1.3#44932: query: www.simulation.net IN A +
04-Sep-2009 19:54:06.330 client 127.0.0.1#39857: query: 6.6.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:06.499 client 127.0.0.1#33561: query: 8.5.168.192.in-addr.arpa IN PTR +
04-Sep-2009 19:54:06.519 client 192.168.1.3#35173: query: www.simulation.net IN A +
04-Sep-2009 19:54:07.662 client 192.168.1.3#44591: query: www.simulation.net IN A +
```

Fig. 5.13 - bindQuery.log file

In the previously given example, the fields of the log file are separated by whitespaces. The fields that can be found are:

- Date,
- Time,
- Client IP address,
- Query of the client.

Finally, let us mention an important note about the log files. In order to reduce the space occupied by these files in the hard drive, when log files reach a certain size the name of the log file, for example log_file, changes to log_file.0 and the new data is stored in a new log_file.

5.2.2. Monitoring the network devices

In order to gather the information of the devices inside the network, the SNMP protocol was configured in the routers (as was explained in the previous chapter). To collect this data on the server, the "Net-SNMP" utility (available in the linux repositories) has to be installed.

This utility allows the communication between SNMP clients and agents. The requests made to the agents are done via terminal so a bash file can be made to do this automatically during the time the emulation is running. The syntax of the command used is shown below:

snmpwalk -v2c -c $community_string $ip IF-MIB::system >> $output

The fields for this command are separated with whitespaces and are described in the following table:

Field	**Description**	**example**
Command	SNMP command (snmpwalk, snmpget and others)	snmpwalk
Protocol version	Can be v1, v2c or v3.	-v2c
Options	-c, -d, -m and many others	-c
Community string	Value of the community string of the equipment.	$community_string
IP	IP address of the equipment	$ip
MIB	Desired MIB by the administrator.	IF-MIB
.1.3.6.1.2.1	.iso.org.dod.internet.mgmt.mib-2.	::
System	Last part of the MIB tree. Together with the previous field, allows the gathering of all the requested data	System
Output file	The output of the command is stored in a .txt file	$output

Table 5.4 - SNMP command fields and description

An example of the output of the "snmpwalk" command is shown in the following figure:

```
IF-MIB::ifInOctets.1 = Counter32: 50118
IF-MIB::ifInOctets.2 = Counter32: 54919
IF-MIB::ifInOctets.3 = Counter32: 117948
IF-MIB::ifInOctets.4 = Counter32: 0
IF-MIB::ifInOctets.5 = Counter32: 0
IF-MIB::ifInUcastPkts.1 = Counter32: 407
IF-MIB::ifInUcastPkts.2 = Counter32: 451
IF-MIB::ifInUcastPkts.3 = Counter32: 1328
IF-MIB::ifInUcastPkts.4 = Counter32: 0
IF-MIB::ifInUcastPkts.5 = Counter32: 0
IF-MIB::ifInNUcastPkts.1 = Counter32: 104
IF-MIB::ifInNUcastPkts.2 = Counter32: 111
IF-MIB::ifInNUcastPkts.3 = Counter32: 8
IF-MIB::ifInNUcastPkts.4 = Counter32: 0
IF-MIB::ifInNUcastPkts.5 = Counter32: 0
```

Fig. 5.14 - snmpwalk output

The script files used to collect the SNMP data are presented in Appendix A.8.

5.2.3. Network Traffic captured in the server

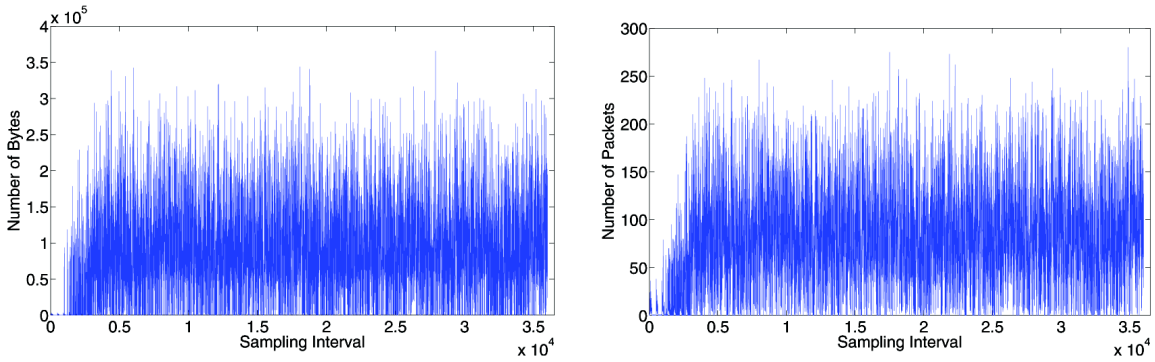


Fig. 5.15 - Aggregated network traffic on the server

The above graphics show the aggregated traffic of the entire network as seen by the server.

The network traffic captured at the server exhibits a profile that is similar to the one that was already explained in sub-section 5.1.

5.3. Summary

This chapter has shown examples of the different data that was gathered to test the correct functioning of the emulation platform. Besides, with the captured network traffic, it is also possible to see the different profiles corresponding to each one of the services that were implemented in the different network scenarios.

Regarding the data collected at the server, the different log files and syntaxes for all the services installed in the server were also explained.

6. Conclusions and Future Work

6.1. Conclusions

This dissertation proved that it is possible to generate real traffic in a closed and controlled emulation environment without having the risk of compromising confidential data. This was achieved by emulating the proposed topologies and connecting the emulation tool to real network devices and a real server.

The virtual interfaces used for the dissertation are proved to act exactly as real network adapters, so it is possible to emulate the behavior of multiple users inside the network, with multiple profiles and using multiple services but, as previously explained in the forth chapter, some measures have to be taken in order to assure that all the virtual interfaces can connect to the network.

The data that was collected in this work only serves as an example of the kind of data that can be gathered in this type of environments and to verify the correct functioning of the emulation platform.

The gathered data will become available on a website that is already made in order to share the results (www.ground-thruth.av.it.pt).

6.2. Future Work

In order to complement this work, there are some interesting topics that can be developed in the future:

- Analyze more complex topologies.
- Emulate more services per interface (P2P file sharing, for example).
- Implement other types of security attacks.
- Study the effects of Spam e-mail.
- Correlate all the gathered data.
- Upgrade the used traffic generation tool in order to generate traffic according also to a daily pattern.
- Improve the website.

Bibliography

Books, articles and other documents

1. Kaufman, Charlie; Perlman, Radia; Speciner, Mike. - (2002) - *Network security: private communication in a public world*. 2nd ed. Upper Saddle River (NJ) : Prentice Hall.
2. Converse, Tim; Park, Joyce; Morgan, Clark. - (2004) - *PHP5 and MySQL bible*. Indianapolis (IN) : Wiley.
3. Mandia, Kevin; Prosise, Chris; Pepe, Matt. - (2003) - *Incident Response & Computer Forensics*. 2nd ed. Mcgraw Hill.
4. Dooley, Kevin; Brown Ian. - (2006) - *Cisco IOS Cookbook.* 2nd ed. O'Reilly Media.
5. Monteiro, Catarina Maria dos Santos Amaral, - Sistema integrado para recolha de informação de rede; sob orientação de Paulo Salvador e António Nogueira. Aveiro : C. Monteiro, 2008. XI, 129 p. : il. color. Tese de mestrado: Engenharia Electrónica e Telecomunicações, Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, 2008.
6. L. Bertolotti and M. C. Calzarossa, "Models of mail server workloads," *Perform. Eval.,* vol. 46, pp. 65-76, 2001. [accessed August 10, 2009].
7. S. W. Neville and L. Kin Fun, "The Rational for Developing Larger-scale 1000+ Machine Emulation-Based Research Test Beds," in *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, 2009, pp. 1092-1099. [accessed July 28, 2009].
8. S. Shah and B. D. Noble, "A study of e-mail patterns," *Softw. Pract. Exper.,* vol. 37, pp. 1515-1538, 2007. [accessed August 10, 2009]
9. L. Song and G. A. Marin, "Realistic Internet traffic simulation through mixture modeling and a case study," in *Simulation Conference, 2005 Proceedings of the Winter*, 2005, p. 9 pp. [accessed July 28, 2009].
10. J. Wang and Y. Hu, "A performance study on Internet Server Provider mail servers," in *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, 2004, pp. 56-61 Vol.1. [accessed August 10, 2009].
11. C. Yen-Wen, "Experimental study of Internet traffic modeling and bandwidth allocation," in *Communications, Computers and signal Processing, 2001. PACRIM.*

2001 IEEE Pacific Rim Conference on, 2001, pp. 587-590 vol.2. [accessed August 11, 2009].

12. PostFix Basic Setup HowTo. Available: https://help.ubuntu.com/community/PostfixBasicSetupHowto

References

[1] P. Barford, et al., "A signal analysis of network traffic anomalies," presented at the Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, Marseille, France, 2002. [Accessed August 15, 2009]

[2] C. Estan, et al., "Automatically inferring patterns of resource consumption in network traffic," presented at the Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, 2003. [Accessed August 15, 2009]

[3] L. H. Gomes, et al., "Characterizing a spam traffic," presented at the Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004. [Accessed August 10, 2009]

[4] A. Lakhina, et al., "Diagnosing network-wide traffic anomalies," presented at the Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, Portland, Oregon, USA, 2004. [Accessed August 14, 2009]

[5] I. Zukerman, et al., "Predicting users' requests on the WWW," presented at the Proceedings of the seventh international conference on User modeling, Banff, Canada, 1999. [Accessed August 1, 2009]

[6] S. Guruprasad, et al., "Integrated network experimentation using simulation and emulation," in Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on, 2005, pp. 204-212. [Accessed August 1, 2009]

[7] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," Networking, IEEE/ACM Transactions on, vol. 9, pp. 392-403, 2001. [Accessed July 21, 2009]

[8] R. R. Kompella, et al., "On scalable attack detection in the network," IEEE/ACM Trans. Netw., vol. 15, pp. 14-25, 2007. [Accessed July 21, 2009]

[9] H. Ringberg, et al., "The need for simulation in evaluating anomaly detectors," SIGCOMM Comput. Commun. Rev., vol. 38, pp. 55-59, 2008. [Accessed July 21, 2009]

[10] Apache. Available: http://httpd.apache.org

[11] Apache Log Files. Available: http://httpd.apache.org/docs/2.2/logs.html

[12] Internet World Stats. Available: http://www.internetworldstats.com

[13] Internet Systems Consortium. Available: http://www.internetworldstats.com

[14] Packet Tracer. Available: http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

[15] The Cnet Simulator. Available: http://www.csse.uwa.edu.au/cnet/

[16] GTNetS. Available: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[17] GTNetS Manual. Available: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/docs/GTNetS_manual.pdf

[18] NCTUns. Available: http://nsl.csie.nctu.edu.tw/nctuns.html

[19] The Netwrok Simulator - NS2. Available: http://www.isi.edu/nsnam/ns/

[20] OMNeT++. Available: http://www.omnetpp.org/

[21] OPNET Technologies, inc. Available: http://www.opnet.com/

[22] Dummynet. Available: http://info.iet.unipi.it/~luigi/dummynet/

[23] Netem. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[24] SIMENA. Available: http://www.simena.net/

[25] Emulab - Network Emulation Testbed. Available: http://www.emulab.net/index.php3?stayhome=1

[26] Common Open Research Emulator - CORE. Available: http://cs.itd.nrl.navy.mil/work/core/index.php

[27] Dynamips. Available: http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

[28] Dynagen. Available: http://www.dynagen.org/

[29] Graphical Network Emulator - GNS3. Available: http://www.gns3.net/

[30] GNS3 Tutorial. Available: http://www.gns3.net/documentation

[31] RFC3164. Available: http://www.ietf.org/rfc/rfc3164.txt

[32] Dynagen Tutorial. Available: http://www.dynagen.org/tutorial.htm

[33] R. Chertov*, et al.*, "Emulation versus simulation: a case study of TCP-targeted denial of service attacks," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 2006, pp. 10 pp.-325. [Accessed July 28, 2009]

[34] RFC 1945. Available: http://www.ietf.org/rfc/rfc1945.txt.

[35] RFC 2616. Available: http://www.ietf.org/rfc/rfc2616.txt.

[36] ProFTPD. Available: http://www.proftpd.org/.

[37] RFC 959. Available: http://tools.ietf.org/html/rfc959.

[38] POSTFIX. Available: http://www.postfix.org/start.html.

[39] Postfix Admin - Web based administration interface. Available: http://postfixadmin.sourceforge.net/.

[40] RFC 821. Available: http://rfc821.com/.

[41] RFC 5321. Available: http://tools.ietf.org/html/rfc5321.

[42] RFC 1939. Available: http://www.ietf.org/rfc/rfc1939.txt.

[43] MySQL. Available: http://www.mysql.com/.

[44] MySQL Workbench. Available: http://wb.mysql.com/.

[45] Bind9 DNS Server. Available: http://www.bind9.net/.

[46] RFC 1034. Available: http://www.ietf.org/rfc/rfc1034.txt.

[47] RFC 1035. Available: http://www.ietf.org/rfc/rfc1035.txt.

[48] Net-SNMP. Available: http://www.net-snmp.org/.

[49] RFC 3416. Available: http://tools.ietf.org/html/rfc3416.

[50] Internetworking Technology Handbook. Available: http://www.cisco.com/en/US/docs/internetworking/technology/handbook/NM-Basics.html.

[51] RFC 1155. Available: http://tools.ietf.org/html/rfc1155.

[52] RFC 2131. Available: http://tools.ietf.org/html/rfc2131.

[53] RFC 2328. Available: http://www.ietf.org/rfc/rfc2328.txt.

[54] Cisco MIB Locator. Available: http://tools.cisco.com/ITDIT/MIBS/MainServlet.

Appendix

Appendix A - Server Configuration

A.1 - Interfaces

The *interfaces* file is found in the */etc/network* directory.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth3
iface eth3 inet static
address 192.168.1.1
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255

auto eth3:1
iface eth3:1 inet static
address 192.168.1.2
netmask 255.255.255.0

auto eth3:2
iface eth3:2 inet static
address 192.168.1.3
netmask 255.255.255.0

auto eth3:3
iface eth3:3 inet static
address 192.168.1.4
netmask 255.255.255.0

auto eth3:4
iface eth3:4 inet static
address 192.168.1.5
netmask 255.255.255.0

auto eth3:5
iface eth3:5 inet static
address 192.168.1.6
netmask 255.255.255.0

auto eth3:6
iface eth3:6 inet static
```

```
address 192.168.1.7
netmask 255.255.255.0

auto eth3:7
iface eth3:7 inet static
address 192.168.1.8
netmask 255.255.255.0

auto eth3:8
iface eth3:8 inet static
address 192.168.1.9
netmask 255.255.255.0

auto eth3:9
iface eth3:9 inet static
address 192.168.1.10
netmask 255.255.255.0
```

Fig. A.1.1 - *interfaces* file

After editing the file, the networking process needs to be restarted so that the modifications can take effect:

```
sudo /etc/init.d/networking restart
```

A.2 - APACHE Configuration and Webpages

```
NameVirtualHost *
<VirtualHost *>
      ServerAdmin webmaster@localhost

      DocumentRoot /var/www/
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
      <Directory /var/www/>
            Options Indexes FollowSymLinks MultiViews
            AllowOverride All
            Order allow,deny
            allow from all
      </Directory>

      ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
      <Directory "/usr/lib/cgi-bin">
            AllowOverride None
            Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
            Order allow,deny
            Allow from all
      </Directory>

      ErrorLog /var/log/apache2/error.log

      # Possible values include: debug, info, notice, warn, error, crit,
      # alert, emerg.
      LogLevel warn

      CustomLog /var/log/apache2/access.log combined
      ServerSignature On

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>

</VirtualHost>
```

Fig. A.2.1 - Modified 000-default file

To restart the APACHE service:

sudo a2enmod rewrite

sudo /etc/init.d/apache2 restart

The webpages used for this work were the Homepages of the CNN, Sky News and BBC News. The following figure shows the three pages hosted in the emulation server.

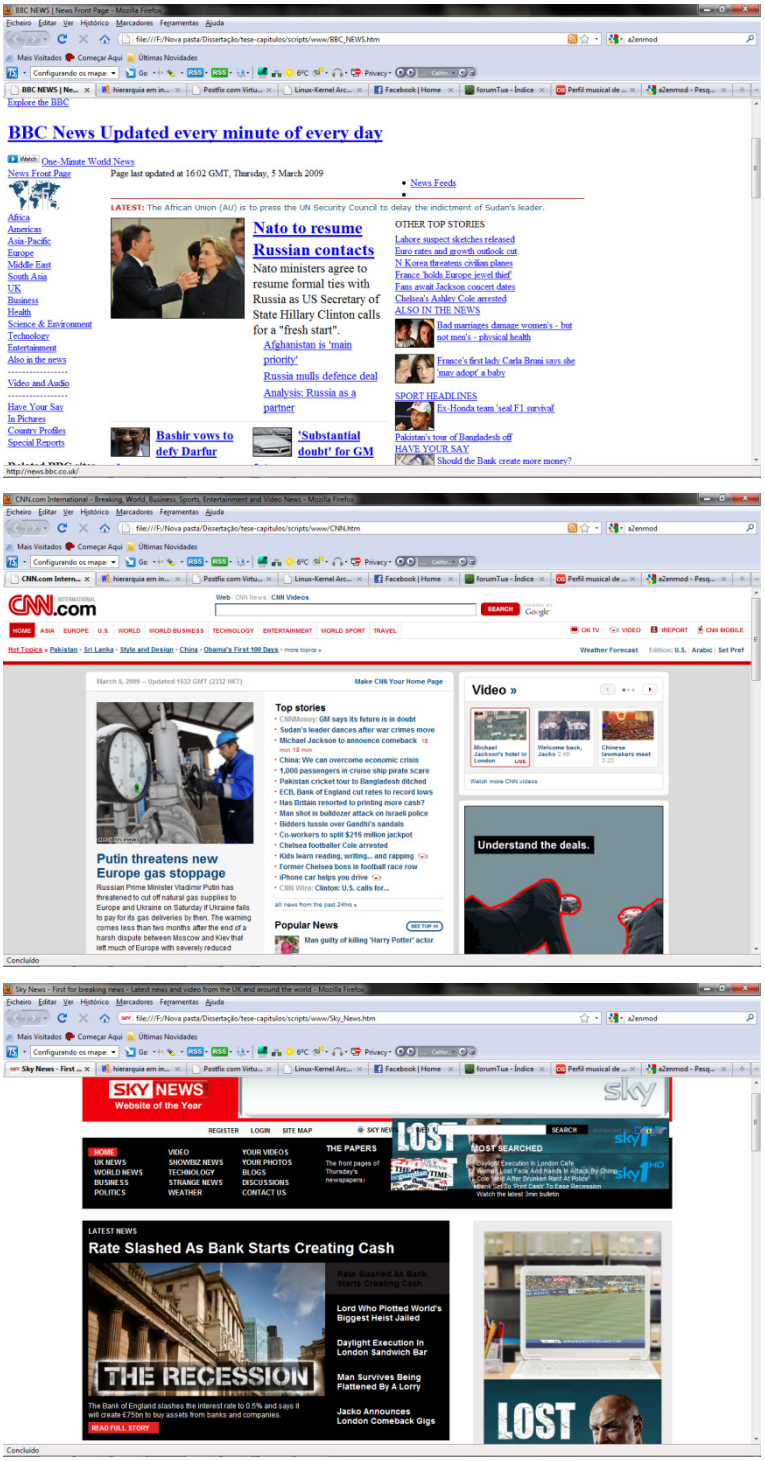


Fig. A.2.2 - webpages kept in the server

A.3 - ProFTPD Configuration

The FTP service used in this work uses user authentication and two specific folders: a download and an upload folder. To configure this service, the following steps were taken:

1. The line "/bin/false/" was added to the /etc/shells file.
2. Creation of the FTP-shared folder:

   Cd /home
   Sudo mkdir FTP-shared

3. Creation of the FTP user:

   sudo useradd userftp -p your_password -d /home/FTP-shared -s /bin/false
   sudo passwd userftp

4. Creation of the upload and download directories and permission settings for the folder:

   cd /home/FTP-shared/
   sudo mkdir download
   sudo mkdir upload
   cd /home
   sudo chmod 755 FTP-shared
   cd FTP-shared
   sudo chmod 755 download
   sudo chmod 777 upload

The *proftpd.conf* file can be found in the /etc/proftp directory. This file keeps information about the FTP user and the folders to be shared. For this work, the *proftpd.conf* file is the following:

```
# To really apply changes reload proftpd after modifications.
AllowOverwrite on
AuthAliasOnly on

# Choose here the user alias you want !!!!
UserAlias sauron userftp

ServerName              "ChezFrodon"
ServerType              standalone
DeferWelcome                  on

MultilineRFC2228 on
DefaultServer                 on
ShowSymlinks                  off

TimeoutNoTransfer 600
TimeoutStalled 100
TimeoutIdle 2200

DisplayFirstChdir               .message
ListOptions                   "-l"

RequireValidShell             off

TimeoutLogin 20

RootLogin               off

# It's better for debug to create log files ;-)
ExtendedLog                   /var/log/ftp.log
TransferLog                   /var/log/xferlog
SystemLog               /var/log/syslog.log

#DenyFilter             \*.*/

# I don't choose to use /etc/ftpusers file (set inside the users you
want to ban, not useful for me)
UseFtpUsers off

# Allow to restart a download
AllowStoreRestart       on

# Port 21 is the standard FTP port, so you may prefer to use another
port for security reasons (choose here the port you want)
Port                    1980

# To prevent DoS attacks, set the maximum number of child processes
# to 30.  If you need to allow more than 30 concurrent connections
# at once, simply increase this value.  Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd)
MaxInstances 8

# Set the user and group that the server normally runs at.
User                  nobody
Group                 nogroup
```

```
# Umask 022 is a good standard umask to prevent new files and dirs
# (second parm) from being group and world writable.
Umask                   022   022

PersistentPasswd        off

MaxClients 8
MaxClientsPerHost 8
MaxClientsPerUser 8
MaxHostsPerUser 8

# Display a message after a successful login
AccessGrantMsg "welcome !!!"
# This message is displayed for each access good or not
ServerIdent                   on       "you're at home"

# Set /home/FTP-shared directory as home directory
DefaultRoot /home/FTP-shared

# Lock all the users in home directory, ***** really important *****
DefaultRoot ~

MaxLoginAttempts    5

#VALID LOGINS
<Limit LOGIN>
AllowUser userftp
DenyALL
</Limit>

<Directory /home/FTP-shared>
Umask 022 022
AllowOverwrite off
      <Limit MKD STOR DELE XMKD RNRF RNTO RMD XRMD>
      DenyAll
      </Limit>
</Directory>

<Directory /home/FTP-shared/download/*>
Umask 022 022
AllowOverwrite off
      <Limit MKD STOR DELE XMKD RNEF RNTO RMD XRMD>
      DenyAll
      </Limit>
</Directory>

<Directory /home/FTP-shared/upload/>
Umask 022 022
AllowOverwrite on
      <Limit READ RMD DELE>
            DenyAll
      </Limit>

      <Limit STOR CWD MKD>
            AllowAll
      </Limit>
</Directory>
```

Fig. A.3.1 - proftpd.conf file

To restart the ProFTPD service:

sudo /etc/init.d/proftpd restart

A.4 - POSTFIX Configuration

Configure the Postfix maps:

sudo vi /etc/postfix/mysql_virtual_alias_maps.cf

```
user = postfix
password = 14478998
hosts = 127.0.0.1
dbname = postfix
table = alias
select_field = goto
where_field = address
```

Fig. A.4.1 - virtual alias map file

sudo vi /etc/postfix/mysql_virtual_domains_maps.cf

```
user = postfix
password = 14478998
hosts = 127.0.0.1
dbname = postfix
table = domain
select_field = domain
where_field = domain
#additional_conditions = and backupmx = '0' and active = '1'
```

Fig. A.4.2 - Virtual domains map file

sudo vi /etc/postfix/mysql_virtual_mailbox_maps.cf

```
user = postfix
password = 14478998
hosts = 127.0.0.1
dbname = postfix
table = mailbox
select_field = maildir
where_field = username
#additional_conditions = and active = '1'
```

Fig. A.4.3 - Virtual mailbox map file

sudo vi /etc/postfix/mysql_virtual_mailbox_limit_maps.cf

```
user = postfix
password = 14478998
hosts = 127.0.0.1
dbname = postfix
table = mailbox
select_field = quota
where_field = username
#additional_conditions = and active = '1'
```

Fig. A.4.4 - virtual mailbos limit map file

sudo vi /etc/postfix/mysql_relay_domains_maps.cf

```
user = postfix
password = 14478998
hosts = 127.0.0.1
dbname = postfix
table = domain
select_field = domain
where_field = domain
additional_conditions = and backupmx = '1'
```

Fig. A.4.5 - relay domains map file

Change file permissions

sudo chgrp postfix /etc/postfix/mysql_*.cf

sudo chmod 640 /etc/postfix/mysql_*.cf

Create the *vmail* user to store the E-mails of all the virtual hosts

sudo groupadd -g 5000 vmail

sudo useradd -m -g vmail -u 5000 -d /home/vmail -s /bin/bash vmail

Postfix configuration

sudo vi /etc/postfix/main.cf

```
# See /usr/share/postfix/main.cf.dist for a commented, more
complete version
```

```
# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# Virtual Mailbox Domain Settings


virtual_alias_maps =
mysql:/etc/postfix/mysql_virtual_alias_maps.cf
virtual_mailbox_domains =
mysql:/etc/postfix/mysql_virtual_domains_maps.cf
virtual_mailbox_maps =
mysql:/etc/postfix/mysql_virtual_mailbox_maps.cf
virtual_mailbox_limit = 51200000
virtual_minimum_uid = 5000
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000
virtual_mailbox_base = /home/vmail
virtual_transport = virtual

# Additional for quota support

virtual_create_maildirsize = yes
virtual_mailbox_extended = yes
virtual_mailbox_limit_maps =
mysql:/etc/postfix/mysql_virtual_mailbox_limit_maps.cf
virtual_mailbox_limit_override = yes
virtual_maildir_limit_message = Sorry, the your maildir has
overdrawn your diskspace quota, please free up some of spaces of
your mailbox try again.
virtual_overquota_bounce = yes

# TLS parameters

smtpd_tls_cert_file = /etc/postfix/smtpd.cert
smtpd_tls_key_file = /etc/postfix/smtpd.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database =
btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database =
```

```
btree:${data_directory}/smtp_scache

#The host name where your MX for virtual domains will point to
myhostname = mta.simulation.net
mydestination = #Remains blank since we are going to host virtual
domains
relayhost =
#Remains blank unless you are going to use your ISP's SMTP server
mail sending out mails.
#In which case it would be set to the host name of the ISP's SMTP
server


alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mynetworks = 192.168.1.0/24 127.0.0.1/32
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all

smtpd_recipient_restrictions = permit_mynetworks,
permit_sasl_authenticated, reject_unauth_destination, permit
# modify the existing smtpd_sender_restrictions
smtpd_sender_restrictions = permit_sasl_authenticated,
permit_mynetworks, reject_unauth_pipelining, permit
# then add these
smtpd_sasl_auth_enable = yes
broken_sasl_auth_clients = yes
smtpd_sasl_path = smtpd
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain =
```

Fig. A.4.6 - Postfix configuration file

IMAP Authentication

sudo vi /etc/courier/authdaemonrc

```
authmodulelist="authmysql authpam"
```

Fig. A.4.7 - Option changed in the authdaemonrc file

sudo vi /etc/courier/authmysqlrc

```
MYSQL_SERVER        127.0.0.1
MYSQL_USERNAME        postfixadmin
MYSQL_PASSWORD        14478998
MYSQL_PORT            0
```

```
MYSQL_OPT            0
MYSQL_DATABASE        postfix
MYSQL_USER_TABLE    mailbox
MYSQL_CRYPT_PWFIELD    password
MYSQL_UID_FIELD        '5000'
MYSQL_GID_FIELD        '5000'
MYSQL_LOGIN_FIELD    username
MYSQL_HOME_FIELD    '/home/vmail'
MYSQL_NAME_FIELD    name
MYSQL_MAILDIR_FIELD    maildir
MYSQL_QUOTA_FIELD    concat(quota,'S')
```

Fig. A.4.8 - authmysqlrc file

Restart the following services:

sudo /etc/init.d/courier-authdaemon restart

sudo /etc/init.d/courier-imap restart

sudo /etc/init.d/courier-imap-ssl restart

sudo /etc/init.d/courier-pop restart

sudo /etc/init.d/courier-pop-ssl restart

sudo tail -f /var/log/mail.info

SMTP Authentication

sudo vi /etc/default/saslauthd

```
START=yes
MECHANISMS="pam"
OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd -r"
```

Fig. A.4.9 - Options needed for SMTP authentication

Start the sasl authentication service:

sudo /etc/init.d/saslauthd start

Password authentication method

sudo vi /etc/postfix/sasl/smtpd.conf

```
pwcheck_method: saslauthd
mech_list: PLAIN LOGIN
log_level: 5
```

Fig. A.4.10 - Options needed to define the type of authentication

SMTP module configuration

sudo vi /etc/pam.d/smtp

```
auth    required   pam_mysql.so user=<user do banco> passwd=<passwd do
mysql> host=127.0.0.1 db=postfix table=mailbox usercolumn=username
passwdcolumn=password crypt=1
account sufficient pam_mysql.so user=<user do banco> passwd=<passwd do
mysql> host=127.0.0.1 db=postfix table=mailbox usercolumn=username
passwdcolumn=password crypt=1
```

Fig. A.4.11 - SMTP module configuration file

Because the SMTP service is used inside a jail, there are some modifications that need to be made:

sudo vi /etc/group

```
sasl:x:45:postfix
```

Fig. A.4.12 - inserted line in the group file to create the sasl group and postif user

Creation of files and directories needed by postfix inside the jail

sudo mkdir -p /var/spool/postfix/var/run/

sudo mv /var/run/saslauthd /var/spool/postfix/var/run/saslauthd

sudo ln -ns /var/spool/postfix/var/run/saslauthd /var/run/saslauthd

sudo cp /etc/sasldb2 /var/spool/postfix/etc/

sudo chgrp sasl /var/spool/postfix/etc/sasldb2

sudo chmod g+w /var/spool/postfix/etc/sasldb2

Restart the authentication and postfix services

sudo /etc/init.d/courier-authdaemon restart

sudo /etc/init.d/saslauthd restart

sudo /etc/init.d/postfix restart

A.5 - PostfixAdmin Configuration

sudo vi /var/www/postfixadmin/config.inc.php

```
$CONF['configured'] = true;

$CONF['database_type'] = 'mysqli';
$CONF['database_host'] = 'localhost';
$CONF['database_user'] = 'postfixadmin';
$CONF['database_password'] = '14478998';
$CONF['database_name'] = 'postfix';

$CONF['postfix_admin_url'] = '';
$CONF['admin_email'] = 'postmaster@change-this-to-your.domain.tld';

$CONF['domain_path'] = 'YES';
$CONF['domain_in_mailbox'] = 'NO';

# Customizações
$CONF['user_footer_link'] = "http://simulation.net/postfixadmin";
$CONF['footer_text'] = 'Return to postfixmyadmin home';
$CONF['footer_link'] = 'http://simulation.net/postfixadmin';

$CONF['welcome_text'] = &lt;&lt;&lt;EOM
Hi,

Welcome to your new account.
EOM;

$CONF['theme_logo'] = 'images/logo-default.png';
$CONF['theme_css'] = 'css/default.css';
```

Fig. A.5.1 - Postfixadmin configuration file

A.6 - MySQL Script

```
#
# Postfix / MySQL
#
CREATE DATABASE postfix;

GRANT SELECT ON postfix.* TO postfix@localhost IDENTIFIED BY
'postfixpassword';
GRANT SELECT, INSERT, DELETE, UPDATE, CREATE, ALTER ON postfix.* TO
postfixadmin@localhost IDENTIFIED BY 'postfixadmin';

USE postfix;
#
# Table structure for table admin
#
CREATE TABLE admin (
username varchar(255) NOT NULL default '',
password varchar(255) NOT NULL default '',
created datetime NOT NULL default '0000-00-00 00:00:00',
modified datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
PRIMARY KEY  (username),
KEY username (username)
) COMMENT='Postfix Admin - Virtual Admins';

#
# Table structure for table alias
#
CREATE TABLE alias (
address varchar(255) NOT NULL default '',
goto text NOT NULL,
domain varchar(255) NOT NULL default '',
created datetime NOT NULL default '0000-00-00 00:00:00',
modified datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
PRIMARY KEY  (address),
KEY address (address)
) COMMENT='Postfix Admin - Virtual Aliases';

#
# Table structure for table domain
#
CREATE TABLE domain (
domain varchar(255) NOT NULL default '',
description varchar(255) NOT NULL default '',
aliases int(10) NOT NULL default '0',
mailboxes int(10) NOT NULL default '0',
maxquota int(10) NOT NULL default '0',
transport varchar(255) default NULL,
backupmx tinyint(1) NOT NULL default '0',
created datetime NOT NULL default '0000-00-00 00:00:00',
modified datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
PRIMARY KEY  (domain),
KEY domain (domain)
) COMMENT='Postfix Admin - Virtual Domains';
```

```
#
# Table structure for table domain_admins
#
CREATE TABLE domain_admins (
username varchar(255) NOT NULL default '',
domain varchar(255) NOT NULL default '',
created datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
KEY username (username)
) COMMENT='Postfix Admin - Domain Admins';

#
# Table structure for table log
#
CREATE TABLE log (
timestamp datetime NOT NULL default '0000-00-00 00:00:00',
username varchar(255) NOT NULL default '',
domain varchar(255) NOT NULL default '',
action varchar(255) NOT NULL default '',
data varchar(255) NOT NULL default '',
KEY timestamp (timestamp)
) COMMENT='Postfix Admin - Log';

#
# Table structure for table mailbox
#
CREATE TABLE mailbox (
username varchar(255) NOT NULL default '',
password varchar(255) NOT NULL default '',
name varchar(255) NOT NULL default '',
maildir varchar(255) NOT NULL default '',
quota int(10) NOT NULL default '0',
domain varchar(255) NOT NULL default '',
created datetime NOT NULL default '0000-00-00 00:00:00',
modified datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
PRIMARY KEY  (username),
KEY username (username)
) COMMENT='Postfix Admin - Virtual Mailboxes';

#
# Table structure for table vacation
#
CREATE TABLE vacation (
email varchar(255) NOT NULL default '',
subject varchar(255) NOT NULL default '',
body text NOT NULL,
cache text NOT NULL,
domain varchar(255) NOT NULL default '',
created datetime NOT NULL default '0000-00-00 00:00:00',
active tinyint(1) NOT NULL default '1',
PRIMARY KEY  (email),
KEY email (email)
) COMMENT='Postfix Admin - Virtual Vacation';
```

Fig. A.6.1 - Database Script for MySQL

A.7 - BIND9 Configuration

To add a Primary Master zone, the file *named.conf.local* found in the bind directory (/etc/bind) has to be edited:

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in
your
// organization
//include "/etc/bind/zones.rfc1918";

# This is the zone definition. replace example.com with your
domain name

zone "simulation.net" {

        type master;

        file "/etc/bind/zones/db.simulation.net";

        };


# This is the zone definition for reverse DNS. replace 0.168.192
with your network address in reverse notation - e.g my network
address is 192.168.0

zone "1.168.192.in-addr.arpa" {

     type master;

     file "/etc/bind/zones/rev.1.168.192.in-addr.arpa";

};

logging {
channel "BindGraphQuery" {
file "/var/log/bindQuery.log";
print-time yes;
severity debug 3;
};
category queries { BindGraphQuery; };
};
```

Fig. A.7.1 - named.conf.local file

The zone "simulation.net" uses a file called *db.simulation.net* to store the zone information:

```
;// replace example.com with your domain name. do not forget the
. after the domain name!
;// Also, replace ns1 with the name of your DNS server
simulation.net.      IN      SOA     sim-server.simulation.net.
admin.simulation.net. (
;
;// Do not modify the following lines!


2006081401

                                                          28800

                                                          3600

                                                          604800

                                                          38400

 )



;// Replace the following line as necessary:

;// sim-server = DNS Server name

;// mta = mail server name

;// example.com = domain name

simulation.net.      IN      NS              sim-
server.simulation.net.
simulation.net.      IN      MX     10       mta.simulation.net.



;// Replace the IP address with the right IP addresses.

www              IN      A       192.168.1.2
mta              IN      A       192.168.1.2
sim-server       IN      A       192.168.1.1
```

Fig. A.7.2 - db.simulation.net file

The reverse zone information for this work is kept in the *rev.1.168.192.in-addr.arpa* file:

```
;//replace example.com with yoour domain name, ns1 with your DNS
server name.

;// The number before IN PTR example.com is the machine address
of the DNS server. in my case, it's 1, as my IP address is
192.168.0.1.

@ IN SOA sim-server.simulation.net. admin.simulation.net. (

                          2006081401;

                          28800;

                          604800;

                          604800;

                          86400

)



                      IN    NS     sim-server.simulation.net.

1                     IN    PTR    simulation.net
```

Fig. A.7.3 - Reverse zone file

In order for the modifications to take effect, the BIND9 service has to be restarted with the following command:

```
/etc/init.d/bind9 restart
```

A.8 - SNMP Script

```
#!/bin/bash
while true; do
      bash /home/sim-client1/Desktop/bash snmp/snmp_b.sh;
      sleep 10;
done
```

Fig. A.8.1 - Example of the control.sh file

```
date >> /home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifInOctets >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifInUcastPkts >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifInNUcastPkts >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifOutOctets >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifOutUcastPkts >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifOutNUcastPkts >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifOutDiscards >>
/home/simulador/Simulation/snmptable_r0.txt
snmpwalk -v2c -c simulation 192.168.2.1 IF-MIB::ifOutErrors >>
/home/simulador/Simulation/snmptable_r0.txt
date >> /home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifInOctets >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifInUcastPkts >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifInNUcastPkts >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifOutOctets >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifOutUcastPkts >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifOutNUcastPkts >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifOutDiscards >>
/home/simulador/Simulation/snmptable_r2.txt
snmpwalk -v2c -c simulation 192.168.2.2 IF-MIB::ifOutErrors >>
/home/simulador/Simulation/snmptable_r2.txt
date >> /home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifInOctets >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifInUcastPkts >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifInNUcastPkts >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifOutOctets >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifOutUcastPkts >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifOutNUcastPkts >>
/home/simulador/Simulation/snmptable_r1.txt
```

```
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifOutDiscards >>
/home/simulador/Simulation/snmptable_r1.txt
snmpwalk -v2c -c simulation 192.168.5.1 IF-MIB::ifOutErrors >>
/home/simulador/Simulation/snmptable_r1.txt
date >> /home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifInOctets >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifInUcastPkts >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifInNUcastPkts >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifOutOctets >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifOutUcastPkts >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifOutNUcastPkts >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifOutDiscards >>
/home/simulador/Simulation/snmptable_r3.txt
snmpwalk -v2c -c simulation 192.168.4.2 IF-MIB::ifOutErrors >>
/home/simulador/Simulation/snmptable_r3.txt
```

Fig. A.8.2 - Example of the snmp_b.sh file

Appendix B - Router Configuration

```
//router setup in GNS3 for scenario_A//

//Router 0//

ena
conf t
hostname Router0
interface f0/0
ip address 192.168.2.1 255.255.255.0
no shut
exit
interface f0/1
ip address 192.168.3.1 255.255.255.0
no shut
exit
interface f1/0
ip address 192.168.1.4 255.255.255.0
no shut
exit
router ospf 1
router-id 1.1.1.1
network 192.168.1.0 0.0.0.255 area 0
network 192.168.2.0 0.0.0.255 area 0
network 192.168.3.0 0.0.0.255 area 0
exit
end
write


//Router 1//

ena
conf t
hostname Router1
interface f0/0
ip address 192.168.2.2 255.255.255.0
no shut
exit
interface f0/1
ip address 192.168.4.1 255.255.255.0
no shut
exit
interface f1/0
ip address 192.168.5.1 255.255.255.0
no shut
exit
router ospf 1
router-id 2.2.2.2
network 192.168.5.0 0.0.0.255 area 0
network 192.168.2.0 0.0.0.255 area 0
network 192.168.4.0 0.0.0.255 area 0
exit
service dhcp
ip dhcp pool 192.168.5.0/24
network 192.168.5.0 255.255.255.0
```

```
dns-server 192.168.1.1
lease 1
exit
ip dhcp excluded-address 192.168.5.1 192.168.5.1
end
write


//Router 2//

ena
conf t
hostname Router2
interface f0/0
ip address 192.168.4.2 255.255.255.0
no shut
exit
interface f0/1
ip address 192.168.3.2 255.255.255.0
no shut
exit
interface f1/0
ip address 192.168.6.1 255.255.255.0
no shut
exit
router ospf 1
router-id 3.3.3.3
network 192.168.6.0 0.0.0.255 area 0
network 192.168.3.0 0.0.0.255 area 0
network 192.168.4.0 0.0.0.255 area 0
exit
service dhcp
ip dhcp pool 192.168.6.0/24
network 192.168.6.0 255.255.255.0
dns-server 192.168.1.1
lease 1
exit
ip dhcp excluded-address 192.168.6.1 192.168.6.1
end
write


//SNMP configuration (after all the routers have their interfaces
configured) //
conf t
snmp-server community test_sim RO
snmp-server ifindex persist
end
write
```

Fig. B.1 - Router configuration for Scenario A

Appendix C - PC Configuration

C.1 - Creation and Configuration of the Interfaces

```
// Creates the Virtual Interfaces//

modprobe tun
tunctl -t tap1
ip l s dev tap1 up
tunctl -t tap2
ip l s dev tap2 up
tunctl -t tap3
ip l s dev tap3 up
tunctl -t tap4
ip l s dev tap4 up
tunctl -t tap5
ip l s dev tap5 up
tunctl -t tap6
ip l s dev tap6 up
tunctl -t tap7
ip l s dev tap7 up
tunctl -t tap8
ip l s dev tap8 up
tunctl -t tap9
ip l s dev tap9 up
tunctl -t tap10
ip l s dev tap10 up
tunctl -t tap11
ip l s dev tap11 up

//Creates the Bridge//

brctl addbr br0
ip l s dev br0 up

//add interfaces to bridge: (TAP and eth0)//

brctl addif br0 tap11
brctl addif br0 eth0

//bridge verification//

brctl show br0

//assign 0/0 address to tap11 and eth0 and static address to br0//

ifconfig tap11 0.0.0.0 promisc up
ifconfig eth0 0.0.0.0 promisc up
ifconfig br0 192.168.1.3/24
```

Fig. C.1.1 - Virtual Interfaces and Bridge configuration for Scenario A

C.2 - IP Routing Table Configuration

```
route add default gw 192.168.5.1 dev tap1
route add default gw 192.168.5.1 dev tap2
route add default gw 192.168.5.1 dev tap3
route add default gw 192.168.5.1 dev tap4
route add default gw 192.168.5.1 dev tap5
route add default gw 192.168.5.1 dev tap12

route add default gw 192.168.6.1 dev tap6
route add default gw 192.168.6.1 dev tap7
route add default gw 192.168.6.1 dev tap8
route add default gw 192.168.6.1 dev tap9
route add default gw 192.168.6.1 dev tap10

ip route add 192.168.5.0/24 dev tap1 table 2
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap1 table 2
ip rule add from 192.168.5.2 table 2
ip rule add to 192.168.5.2 table 2

ip route add 192.168.5.0/24 dev tap2 table 3
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap2 table 3
ip rule add from 192.168.5.3 table 3
ip rule add to 192.168.5.3 table 3

ip route add 192.168.5.0/24 dev tap3 table 4
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap3 table 4
ip rule add from 192.168.5.4 table 4
ip rule add to 192.168.5.4 table 4

ip route add 192.168.5.0/24 dev tap4 table 5
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap4 table 5
ip rule add from 192.168.5.5 table 5
ip rule add to 192.168.5.5 table 5

ip route add 192.168.5.0/24 dev tap5 table 6
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap5 table 6
ip rule add from 192.168.5.6 table 6
ip rule add to 192.168.5.6 table 6

ip route add 192.168.6.0/24 dev tap6 table 7
ip route add 0.0.0.0/0 via 192.168.6.1 dev tap6 table 7
ip rule add from 192.168.6.2 table 7
ip rule add to 192.168.6.2 table 7

ip route add 192.168.6.0/24 dev tap7 table 8
ip route add 0.0.0.0/0 via 192.168.6.1 dev tap7 table 8
ip rule add from 192.168.6.3 table 8
ip rule add to 192.168.6.3 table 8

ip route add 192.168.6.0/24 dev tap8 table 9
ip route add 0.0.0.0/0 via 192.168.6.1 dev tap8 table 9
ip rule add from 192.168.6.4 table 9
ip rule add to 192.168.6.4 table 9

ip route add 192.168.6.0/24 dev tap9 table 10
ip route add 0.0.0.0/0 via 192.168.6.1 dev tap9 table 10
```

```
ip rule add from 192.168.6.5 table 10
ip rule add to 192.168.6.5 table 10

ip route add 192.168.6.0/24 dev tap10 table 11
ip route add 0.0.0.0/0 via 192.168.6.1 dev tap10 table 11
ip rule add from 192.168.6.6 table 11
ip rule add to 192.168.6.6 table 11

ip route add 192.168.5.0/24 dev tap12 table 12
ip route add 0.0.0.0/0 via 192.168.5.1 dev tap12 table 12
ip rule add from 192.168.5.7 table 12
ip rule add to 192.168.5.7 table 12


echo 1 > /proc/sys/net/ipv4/conf/tap1/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap2/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap3/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap4/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap5/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap6/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap7/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap8/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap9/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap10/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/tap12/arp_filter
```

Fig. C.2.1 - PC configuration for Scenario A

Appendix D - Scenarios

D.1 - Scenario A

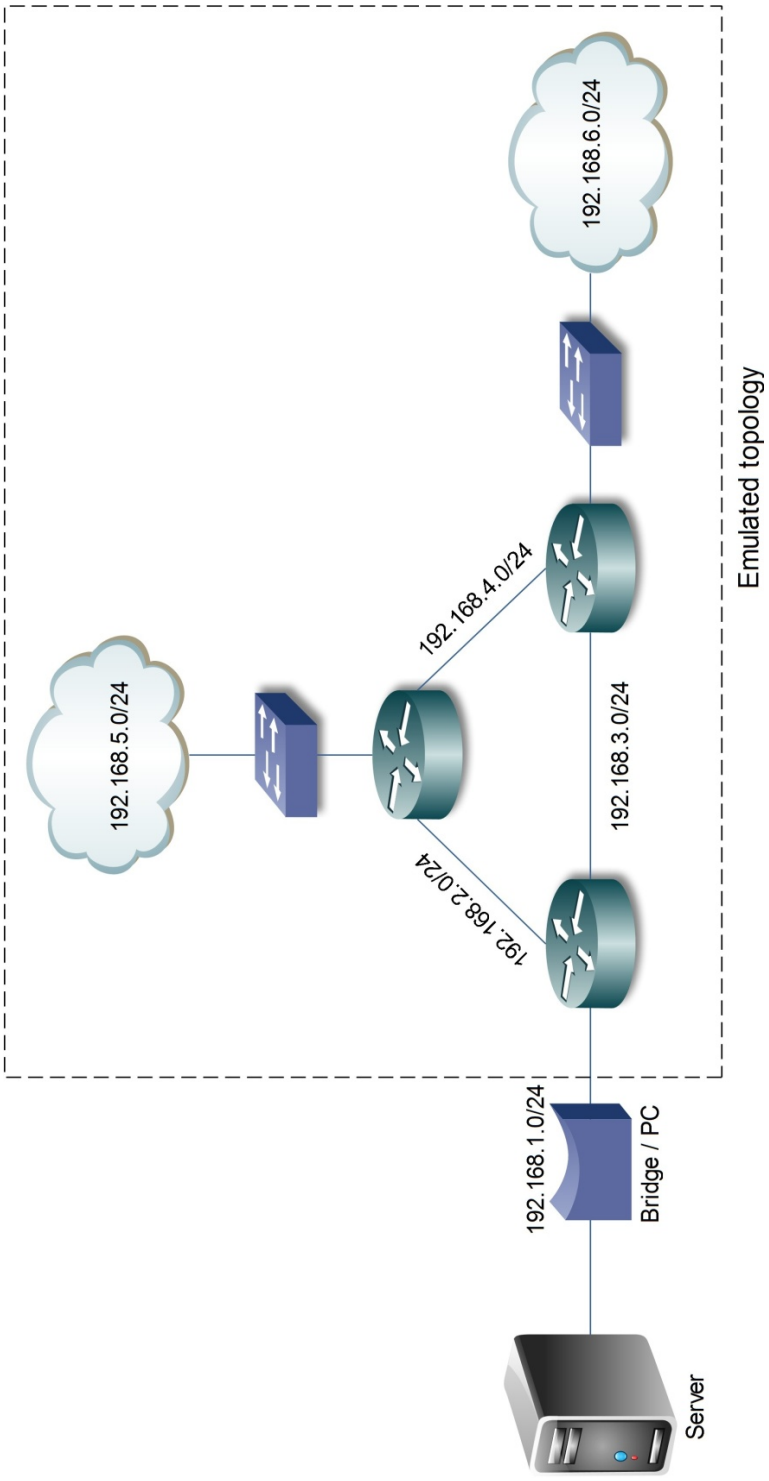


Fig. D.1.1 - Scenario A

D.2 - Scenario B

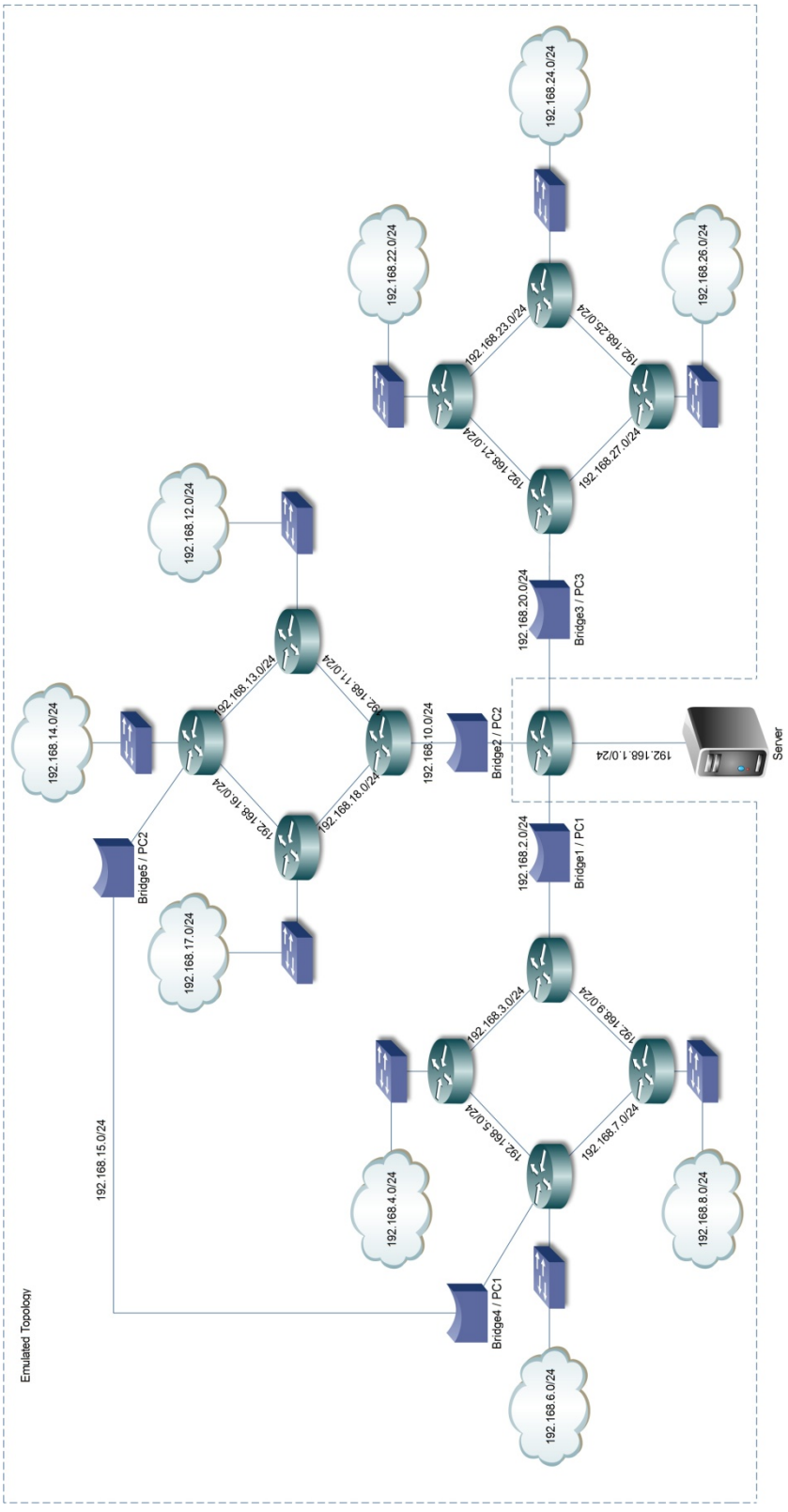


Fig. D.2.1 - Scenario B

D.3 - Scenario C

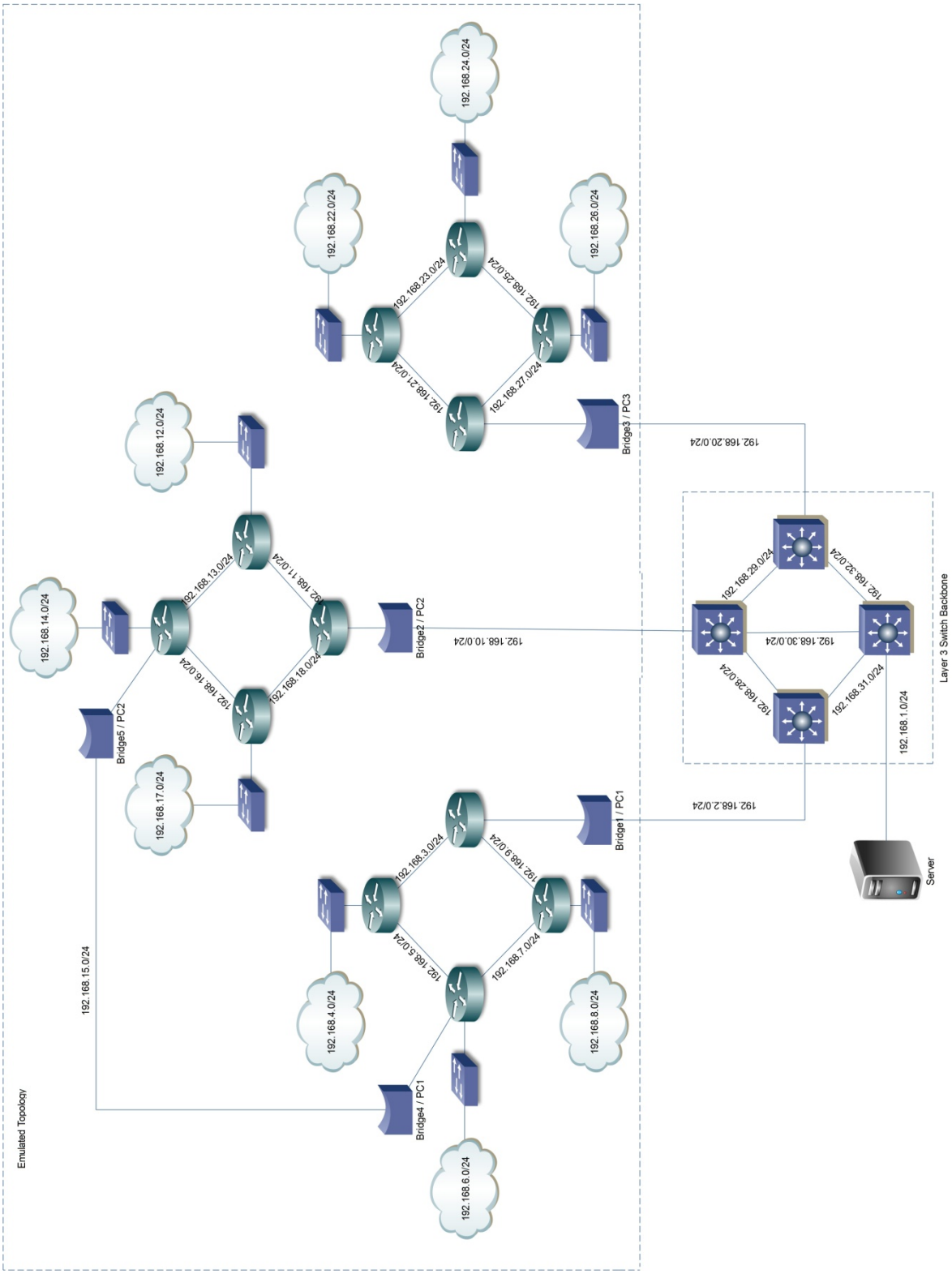


Fig. D.3.1 - Scenario C

Appendix E - Shell Scripts

Shell is a Linux command interpreter. Because it is an interpreted language it does not need to be compiled to work. Among all the existing Shells, the Born Again shell (bash) is the mostly used.

The Bourne Again Shell was written by Stephen Bourne and it was the default and only shell in the UNIX operative systems.

The commands can be sent to the interpreter in two different ways:

- Interactive: the commands are inserted in the command line and passed, one by one, to the interpreter.
- Non interactive: the commands are passed to the interpreter via a shell script file. The shell script files are simple text-based files containing a sequence of commands to be executed.

The first line of a script file indicates the name of the shell that is going to interpret the script. For this work, the bash interpreter was used so the first line of instruction is **#!bin/bash**.

It is also necessary that the script is an executable file. To do this, is used the **chmod +x name_script** command.

To execute a script, the following commands can be used:

Command	Condition
./script_name	If the script is found in the current directory.
$path/script_name	If the scrip is not in the current directory

Table E.1 - Commands to execute a script file

The script files used for this work can be seen in the following figures:

```
#!bin/bash
wget --bind-address=192.168.4.2 www.simulation.net/CNN.htm
```

Fig. E.1 - get.sh file for HTTP request

```
#!bin/bash
curl --interface tap1 ftp://www.simulation.net/download/Screenshot.png --user simulation:simulation -o screenshot.png
```

Fig. E.2 - get.sh file for FTP download

```
#!/bin/bash
curl --interface tap2 ftp://www.simulation.net/download/Screenshot.png --user simulation:simulation -o screenshot.png
```

Fig. E.3 - get.sh file for FTP upload

```
#!/bin/bash

(sleep 15
echo "ehlo mta.simulation.net"
sleep 1
echo "mail from: conta1@simulation.net"
sleep 1
echo "rcpt to: conta16@simulation.net"
sleep 1
echo "DATA"
sleep 1
echo -e "Subject: -test-\n\n"
sleep 1
num=`cat mailnum`
if [ $num -lt 2 ]; then
echo 2 > mailnum
cat smallmail.txt
else
echo 1 > mailnum
cat bigmail.txt
fi
echo "."
sleep 1
echo quit)| netcat -s 192.168.4.2 -vvv mta.simulation.net 25
```

Fig. E.4 - make_mail.sh file

```
#!/bin/bash

(sleep 10
echo user conta1@simulation.net
sleep 10
echo pass 14478998
sleep 10
echo stat
sleep 10
num=0
while true
do
sleep 20
if [ -s mailfile ]
then
num=`tail -1 mailfile | cut -d" " -f2 | sed 's/[^0-9]//g'`
break
else continue
fi
done
echo list
sleep 10
```

```
j=1
until [ $j -gt $num ]
do
echo -n $j >> numfile
echo -n " " >> numfile
j=`expr $j + 1`
if [ $j -gt 15 ]
then break
fi
done
sleep 10
for i in `cat numfile`
do
echo retr $i 0
sleep 5
done
for i in `cat numfile`
do
echo dele $i 0
sleep 5
done
echo quit)| netcat -s 192.168.4.2 -vvv mta.simulation.net 110 > mailfile
rm numfile
```

Fig. E.5 - see_mail.sh file

```
#! /bin/bash
while true; do
      date >> /media/disk/Scenario_A/nmap/1/aggressive-no-source-
port/tap12/nmap_info.txt;
      nmap -sV -v -S 192.168.5.7 -T4 -O www.simulation.net >>
/media/disk/Scenario_A/nmap/1/aggressive-no-source-
port/tap12/nmap_info.txt;
      sleep 60;
done
```

Fig. E.6 - nmap_aggressive.sh file

```
#! /bin/bash
while true; do
      date >> /media/disk/Scenario_A/nmap/1/Normal-
nsp/tap12/nmap_info.txt;
      nmap -S 192.168.5.7 -T3 -sV -v -O www.simulation.net >>
/media/disk/Scenario_A/nmap/1/Normal-nsp/tap12/nmap_info.txt;
      sleep 60;
done
```

Fig. E.7 - nmap_normal.sh file

```
#! /bin/bash
while true; do
      date >> /media/disk/Scenario_A/nmap/1/Sneaky-no-source-
port/tap12/nmap_info.txt;
      nmap -S 192.168.5.7 -T1 -sV -v -O www.simulation.net >>
/media/disk/Scenario_A/nmap/1/Sneaky-no-source-port/tap12/nmap_info.txt;
      sleep 60;
done
```

Fig. E.8 - nmap_sneaky.sh file

```
#!/bin/bash
curl --interface tap12 -T snapshot_pointer.png
ftp://www.simulation.net/upload/ --user simulation:simulation
```

Fig. E.9 - snap-exp.sh file

```
#!/bin/bash
curl --interface tap12 -T snapshot_desk.png
ftp://www.simulation.net/upload/ --user simulation:simulation
```

Fig. E.10 - snap-per.sh file

Appendix F - M-Files

```
function [statistic]=read_data(data,num)

%load data;

[x,s] = size(data);

for i=1:x
    statistic(i) = data(i,num);
end

plot(statistic);

xlabel('Sampling Interval')

ylabel('Number of Bytes')

return
```

Fig. F.1 - M-file used to generate the profile graphics by the number of Bytes

```
function [statistic]=read_data2(data,num)

%load data;

[x,s] = size(data);

for i=1:x
    statistic(i) = data(i,num);
end

plot(statistic);

xlabel('Sampling Interval')

ylabel('Number of Packets')

return
```

Fig. F.2 - M-file used to generate the profile graphics by the number of packets

Appendix G - Network Devices supported MIBs

Cisco 3745 Router [54]	**Cisco 3825 Router [54]**	**Cisco Catalyst 3750ME [54]**
ADSL-DMT-LINE-MIB	ADSL-DMT-LINE-MIB	BGP4-MIB (RFC1657)
ADSL-LINE-MIB	ADSL-LINE-MIB	BRIDGE-MIB (RFC1493)
ATM-MIB	ATM-MIB	CISCO-BULK-FILE-MIB
BGP4-MIB	BGP4-MIB	CISCO-CDP-MIB
BRIDGE-MIB	BRIDGE-MIB	CISCO-CLASS-BASED-QOS-MIB
CISCO-AAA-SERVER-MIB	CISCO-AAA-SERVER-MIB	CISCO-CLUSTER-MIB
CISCO-AAA-SESSION-MIB	CISCO-AAA-SESSION-MIB	CISCO-CONFIG-COPY-MIB
CISCO-AAL5-MIB	CISCO-AAL5-MIB	CISCO-CONFIG-MAN-MIB
CISCO-ACCESS-ENVMON-MIB	CISCO-ACCESS-ENVMON-MIB	CISCO-ENVMON-MIB
CISCO-ADSL-DMT-LINE-MIB	CISCO-ADSL-DMT-LINE-MIB	CISCO-FLASH-MIB
CISCO-ATM-EXT-MIB	CISCO-ATM-EXT-MIB	CISCO-FTP-CLIENT-MIB
CISCO-ATM-PVCTRAP-EXTN-MIB	CISCO-ATM-PVCTRAP-EXTN-MIB	CISCO-HSRP-MIB
CISCO-BGP4-MIB	CISCO-BGP4-MIB	CISCO-HSRP-EXT-MIB
CISCO-BULK-FILE-MIB	CISCO-BULK-FILE-MIB	CISCO-IGMP-FILTER-MIB
CISCO-BUS-MIB	CISCO-BUS-MIB	CISCO-IMAGE-MIB
CISCO-CALL-APPLICATION-MIB	CISCO-CALL-APPLICATION-MIB	CISCO-L2L3-INTERFACE-CONFIG-MIB
CISCO-CALL-HISTORY-MIB	CISCO-CALL-HISTORY-MIB	CISCO-LAG-MIB
CISCO-CAR-MIB	CISCO-CAR-MIB	CISCO-MAC-NOTIFICATION-MIB
CISCO-CAS-IF-MIB	CISCO-CAS-IF-MIB	CISCO-MEMORY-POOL-MIB
CISCO-CCME-MIB	CISCO-CCME-MIB	CISCO-PAGP-MIB
CISCO-CDP-MIB	CISCO-CDP-MIB	CISCO-PING-MIB
CISCO-CIRCUIT-INTERFACE-MIB	CISCO-CIRCUIT-INTERFACE-MIB	CISCO-PORT-SECURITY-MIB
CISCO-CLASS-BASED-	CISCO-CLASS-BASED-	CISCO-PROCESS-MIB

QOS-MIB	QOS-MIB	
CISCO-COMPRESSION-SERVICE-ADAPTER-MIB	CISCO-COMPRESSION-SERVICE-ADAPTER-MIB	CISCO-RTTMON-MIB
CISCO-CONFIG-COPY-MIB	CISCO-CONFIG-COPY-MIB	CISCO-STACK-MIB
CISCO-CONFIG-MAN-MIB	CISCO-CONFIG-MAN-MIB	CISCO-STP-EXTENSIONS-MIB
CISCO-DATA-COLLECTION-MIB	CISCO-DATA-COLLECTION-MIB	CISCO-SYSLOG-MIB
CISCO-DIAL-CONTROL-MIB	CISCO-DDP-IAPP-MIB	CISCO-TCP-MIB
CISCO-DOT11-CONTEXT-SERVICES-MIB	CISCO-DIAL-CONTROL-MIB	CISCO-VLAN-IFTABLE-RELATIONSHIP-MIB
CISCO-DSL-CPE-MIB	CISCO-DOT11-ASSOCIATION-MIB	CISCO-VLAN-MEMBERSHIP-MIB
CISCO-DSP-MGMT-MIB	CISCO-DOT11-CONTEXT-SERVICES-MIB	CISCO-VTP-MIB
CISCO-EMBEDDED-EVENT-MGR-MIB	CISCO-DOT11-IF-MIB	ENTITY-MIB (RFC2737)
CISCO-ENTITY-ASSET-MIB	CISCO-DSL-CPE-MIB	ETHERLIKE-MIB (RFC1398)
CISCO-ENTITY-EXT-MIB	CISCO-DSP-MGMT-MIB	IEEE8023-LAG-MIB
CISCO-ENTITY-VENDORTYPE-OID-MIB	CISCO-EMBEDDED-EVENT-MGR-MIB	IF-MIB (RFC1573)
CISCO-ENVMON-MIB	CISCO-ENTITY-ASSET-MIB	IGMP-MIB
CISCO-FLASH-MIB	CISCO-ENTITY-EXT-MIB	IPMROUTE-MIB
CISCO-FRAME-RELAY-MIB	CISCO-ENTITY-VENDORTYPE-OID-MIB	OSPF-MIB (RFC1253)
CISCO-FTP-CLIENT-MIB	CISCO-ENVMON-MIB	OLD-CISCO-CHASSIS-MIB
CISCO-GPRS-GTP-MIB	CISCO-FLASH-MIB	OLD-CISCO-INTERFACES-MIB
CISCO-H323-TC-MIB	CISCO-FRAME-RELAY-MIB	OLD-CISCO-IP-MIB
CISCO-HSRP-EXT-MIB	CISCO-FTP-CLIENT-MIB	OLD-CISCO-SYS-MIB
CISCO-HSRP-MIB	CISCO-GPRS-GTP-MIB	OLD-CISCO-TS-MIB
CISCO-ICSUDSU-MIB	CISCO-H323-TC-MIB	PIM-MIB
CISCO-IETF-ATM2-PVCTRAP-MIB	CISCO-HSRP-EXT-MIB	MIB-II (RFC1213)

CISCO-IETF-ATM2-PVCTRAP-MIB-EXTN	CISCO-HSRP-MIB	RMON-MIB (RFC1757)
CISCO-IETF-IP-FORWARD-MIB	CISCO-ICSUDSU-MIB	RMON2-MIB (RFC2021)
CISCO-IETF-IP-MIB	CISCO-IETF-ATM2-PVCTRAP-MIB	SNMP-FRAMEWORK-MIB (RFC2571)
CISCO-IETF-NAT-MIB	CISCO-IETF-ATM2-PVCTRAP-MIB-EXTN	SNMP-MPD-MIB (RFC2572)
CISCO-IF-EXTENSION-MIB	CISCO-IETF-DOT11-QOS-EXT-MIB	SNMP-NOTIFICATION-MIB (RFC2573)
CISCO-IMAGE-MIB	CISCO-IETF-DOT11-QOS-MIB	SNMP-TARGET-MIB (RFC2573)
CISCO-IP-STAT-MIB	CISCO-IETF-IP-FORWARD-MIB	SNMP-VACM-MIB (SNMP-VIEW-BASED-ACM-MIB) (RFC2575)
CISCO-IPMROUTE-MIB	CISCO-IETF-IP-MIB	SNMP-USM-MIB (SNMP-USER-BASED-SM-MIB) (RFC2574)
CISCO-IPSEC-FLOW-MONITOR-MIB	CISCO-IETF-NAT-MIB	SNMPv2-MIB (RFC1907)
CISCO-IPSEC-MIB	CISCO-IF-EXTENSION-MIB	TCP-MIB (RFC2012)
CISCO-IPSEC-POLICY-MAP-MIB	CISCO-IMAGE-MIB	UDP-MIB (RFC2013)
CISCO-ISDN-MIB	CISCO-IP-STAT-MIB	CISCO-PAE-MIB
CISCO-ISDNU-IF-MIB	CISCO-IPMROUTE-MIB	CISCO-PRIVATE-VLAN-MIB
CISCO-LEC-DATA-VCC-MIB	CISCO-IPSEC-FLOW-MONITOR-MIB	CISCO-PORTSTORM-CONTROL-MIB
CISCO-LEC-EXT-MIB	CISCO-IPSEC-MIB	CISCO-UDLDP-MIB
CISCO-LECS-MIB	CISCO-IPSEC-POLICY-MAP-MIB	IEEE8021-PEA-MIB
CISCO-LES-MIB	CISCO-ISDN-MIB	MPLS-LDP-MIB
CISCO-MEMORY-POOL-MIB	CISCO-ISDNU-IF-MIB	MPLS-LSR-MIB
CISCO-MOBILE-IP-MIB	CISCO-L2-DEV-MONITORING-MIB	MPLS-VPN-MIB
CISCO-MVPN-MIB	CISCO-LEC-DATA-VCC-	CISCO-PORT-QOS-MIB

	MIB	
CISCO-NBAR-PROTOCOL-DISCOVERY-MIB	CISCO-LEC-EXT-MIB	CISCO-DHCP-SNOOPING-MIB
CISCO-NETFLOW-MIB	CISCO-LECS-MIB	CISCO-BRIDGE-EXT-MIB
CISCO-NTP-MIB	CISCO-LES-MIB	CISCO-IETF-IP-MIB
CISCO-OSPF-MIB	CISCO-MEMORY-POOL-MIB	CISCO-IETF-IP-FORWARD-MIB
CISCO-OSPF-TRAP-MIB	CISCO-MOBILE-IP-MIB	
CISCO-PIM-MIB	CISCO-MVPN-MIB	
CISCO-PING-MIB	CISCO-NBAR-PROTOCOL-DISCOVERY-MIB	
CISCO-POP-MGMT-MIB	CISCO-NETFLOW-MIB	
CISCO-PPPOE-MIB	CISCO-NTP-MIB	
CISCO-PROCESS-MIB	CISCO-OSPF-MIB	
CISCO-QUEUE-MIB	CISCO-OSPF-TRAP-MIB	
CISCO-RAS-MIB	CISCO-PIM-MIB	
CISCO-RF-MIB	CISCO-PING-MIB	
CISCO-RMON-CONFIG-MIB	CISCO-POP-MGMT-MIB	
CISCO-RMON-SAMPLING-MIB	CISCO-PPPOE-MIB	
CISCO-RTTMON-MIB	CISCO-PROCESS-MIB	
CISCO-SAA-APM-MIB	CISCO-QUEUE-MIB	
CISCO-SIP-UA-MIB	CISCO-RAS-MIB	
CISCO-SMI	CISCO-RF-MIB	
CISCO-SNAPSHOT-MIB	CISCO-RMON-CONFIG-MIB	
CISCO-SNMP-TARGET-EXT-MIB	CISCO-RMON-SAMPLING-MIB	
CISCO-SRST-MIB	CISCO-RTTMON-MIB	
CISCO-STACKMAKER-MIB	CISCO-SAA-APM-MIB	
CISCO-SYSLOG-MIB	CISCO-SIP-UA-MIB	
CISCO-TC	CISCO-SMI	
CISCO-TCP-MIB	CISCO-SNAPSHOT-MIB	
CISCO-VLAN-IFTABLE-RELATIONSHIP-MIB	CISCO-SNMP-TARGET-EXT-MIB	

CISCO-VLAN-MEMBERSHIP-MIB	CISCO-SRST-MIB	
CISCO-VOICE-ANALOG-IF-MIB	CISCO-STACKMAKER-MIB	
CISCO-VOICE-ATM-DIAL-CONTROL-MIB	CISCO-SYSLOG-MIB	
CISCO-VOICE-COMMON-DIAL-CONTROL-MIB	CISCO-TBRIDGE-DEV-IF-MIB	
CISCO-VOICE-DIAL-CONTROL-MIB	CISCO-TC	
CISCO-VOICE-DNIS-MIB	CISCO-TCP-MIB	
CISCO-VOICE-ENABLED-LINK-MIB	CISCO-VLAN-IFTABLE-RELATIONSHIP-MIB	
CISCO-VOICE-FR-DIAL-CONTROL-MIB	CISCO-VLAN-MEMBERSHIP-MIB	
CISCO-VOICE-IF-MIB	CISCO-VOICE-ANALOG-IF-MIB	
CISCO-VOICE-LMR-MIB	CISCO-VOICE-ATM-DIAL-CONTROL-MIB	
CISCO-VOICE-NUMBER-EXPANSION-MIB	CISCO-VOICE-COMMON-DIAL-CONTROL-MIB	
CISCO-VOICE-URI-CLASS-MIB	CISCO-VOICE-DIAL-CONTROL-MIB	
CISCO-VPDN-MGMT-EXT-MIB	CISCO-VOICE-DNIS-MIB	
CISCO-VPDN-MGMT-MIB	CISCO-VOICE-ENABLED-LINK-MIB	
CISCO-VSIMASTER-MIB	CISCO-VOICE-FR-DIAL-CONTROL-MIB	
CISCO-VTP-MIB	CISCO-VOICE-IF-MIB	
CISCO-WRED-MIB	CISCO-VOICE-LMR-MIB	
DIAL-CONTROL-MIB	CISCO-VOICE-NUMBER-EXPANSION-MIB	
DS1-MIB	CISCO-VOICE-URI-CLASS-MIB	
DS3-MIB	CISCO-VPDN-MGMT-EXT-	

	MIB	
ENTITY-MIB	CISCO-VPDN-MGMT-MIB	
ETHERLIKE-MIB	CISCO-VSIMASTER-MIB	
EVENT-MIB	CISCO-VTP-MIB	
EXPRESSION-MIB	CISCO-WLAN-VLAN-MIB	
FUNI-MIB	CISCO-WRED-MIB	
HC-RMON-MIB	DIAL-CONTROL-MIB	
HDSL2-SHDSL-LINE-MIB	DS1-MIB	
IF-MIB	DS3-MIB	
IGMP-STD-MIB	ENTITY-MIB	
IMA-MIB	ETHERLIKE-MIB	
INT-SERV-GUARANTEED-MIB	EVENT-MIB	
INT-SERV-MIB	EXPRESSION-MIB	
IP-FORWARD-MIB	FUNI-MIB	
IPMROUTE-STD-MIB	HC-RMON-MIB	
ISDN-MIB	HDSL2-SHDSL-LINE-MIB	
LAN-EMULATION-CLIENT-MIB	IEEE802dot11-MIB	
MIP-MIB	IF-MIB	
MPLS-LDP-MIB	IGMP-STD-MIB	
MPLS-LSR-MIB	IMA-MIB	
MPLS-TE-MIB	INT-SERV-GUARANTEED-MIB	
MPLS-VPN-MIB	INT-SERV-MIB	
MSDP-MIB	IP-FORWARD-MIB	
OLD-CISCO-CHASSIS-MIB	IPMROUTE-STD-MIB	
OLD-CISCO-CPU-MIB	ISDN-MIB	
OLD-CISCO-FLASH-MIB	LAN-EMULATION-CLIENT-MIB	
OLD-CISCO-INTERFACES-MIB	MIP-MIB	
OLD-CISCO-IP-MIB	MPLS-LDP-MIB	
OLD-CISCO-MEMORY-MIB	MPLS-LSR-MIB	
OLD-CISCO-SYSTEM-MIB	MPLS-TE-MIB	

OLD-CISCO-TCP-MIB	MPLS-VPN-MIB	
OLD-CISCO-TS-MIB	MSDP-MIB	
OSPF-MIB	OLD-CISCO-CHASSIS-MIB	
OSPF-TRAP-MIB	OLD-CISCO-CPU-MIB	
PIM-MIB	OLD-CISCO-FLASH-MIB	
Q-BRIDGE-MIB	OLD-CISCO-INTERFACES-MIB	
RFC1213-MIB	OLD-CISCO-IP-MIB	
RFC1231-MIB	OLD-CISCO-MEMORY-MIB	
RFC1315-MIB	OLD-CISCO-SYSTEM-MIB	
RFC1381-MIB	OLD-CISCO-TCP-MIB	
RFC1382-MIB	OLD-CISCO-TS-MIB	
RFC1406-MIB	OSPF-MIB	
RMON-MIB	OSPF-TRAP-MIB	
RMON2-MIB	PIM-MIB	
RS-232-MIB	Q-BRIDGE-MIB	
RSVP-MIB	RFC1213-MIB	
SMON-MIB	RFC1231-MIB	
SNMP-NOTIFICATION-MIB	RFC1315-MIB	
SNMP-PROXY-MIB	RFC1381-MIB	
SNMP-TARGET-MIB	RFC1382-MIB	
SNMP-USM-MIB	RFC1406-MIB	
SNMP-VACM-MIB	RMON-MIB	
SNMPv2-MIB	RMON2-MIB	
SONET-MIB	RS-232-MIB	
SOURCE-ROUTING-MIB	RSVP-MIB	
TCP-MIB	SMON-MIB	
UDP-MIB	SNMP-NOTIFICATION-MIB	
VRRP-MIB	SNMP-PROXY-MIB	
XGCP-MIB	SNMP-TARGET-MIB	
	SNMP-USM-MIB	
	SNMP-VACM-MIB	
	SNMPv2-MIB	
	SONET-MIB	
	SOURCE-ROUTING-MIB	

	TCP-MIB	
	UDP-MIB	
	VRRP-MIB	
	XGCP-MIB	

Table G.2 - List of supported MIBs