Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica

Classe L-31

IDS/IPS: Intrusion Detection/Prevention System

Studente:

Rando Veizi

Relatore:

prof. Fausto Marcantoni

Anno academico: 2012-2013

Table of Contents

Abstract

Attraverso questa tesi vi mostrerò i passi per fare un sistema sicuro basato interamente su software open source.

In particolare, mostreremo come evitare intrusioni dall'esterno attraverso l'utilizzo di sistemi IDS / IPS.

I vari programmi che abbiamo installato avranno il compito di rilevare e bloccare i tentativi di malintenzionati di entrare nel sistema implementato. I diversi attacchi verranno catalogati in un database per renderli innocui.

Lo scopo principale di questo lavoro è di illustrare l'operatività di questo sistema, per mostrare in dettaglio come il software che abbiamo usato lavori per raggiungere il livello di sicurezza auspicato.

Acknowledgments

I WOULD LIKE TO THANK MY SENIOR SUPERVISOR,
MARCANTONI FAUSTO,
FOR HIS SUPPORT AND GUIDANCE
THROUGHOUT THIS MEAL.

Introduction

After the birth of the Internet the way people communicate has changed dramatically, adding a new dimension.

This change has affected not only the communication between individuals, but also the relationship of a company with its customers.

These days companies and organizations need to communicate with their customers through a internet connection, and since those companies have capitals that can be in stake the internet connetion must be secure and protected.

Very often companies spend huge amounts of money for the construction of secure systems so they canprevent attacks from outside or from unauthorized parties.

Through this thesis I will show you the steps to make a secure system based entirely on open source software.

In particular, we will show how to avoid intrusion from the outside through the use of IDS/IPS systems.

The various programs that we installed will have the task to detect and block malicious attempts from entering in the implemented system. The different attacks will be cataloged in a database in order to render them harmless.

For the configuration we will use graphical interfaces that allow the system administrator to manage better, easily and more quickly, the network that we have configured.

It will be easyer to read log files, inside of which will be described in a detail, those events that threaten the integrity of the system.

The main purpose of this work is to illustrate the operationality of this system, to show in detail how the software that we used works to achieve the level of security sought.

IPS/IDS Systems

What are those systems anyway?

Why do we need them?

What pros and cons do they have?

IDS watches a copy of the traffic, IPS watches the real traffic. So, I you want to be alerted of situations, and not affect real traffic, IDS may be for you. Problem here is that since IDS is only watching copied traffic, and alerting you on that, the real offending packet(s) have already passed to their intended target. Even if you have your IDS setup to update your firewall with blocking rules, the initial attack packet has already gone through. If you want to block an atomic attack (single packet attacked) that will get though the IDS, then maybe you should consider the IPS.

Let's talk about risk.

The IPS technology tends to bring more risk to the table then the IDS. The reasoning for this is that the IPS has live traffic passing through it. So if IPS fails, or becomes overwhelmed, it will affect your live traffic. This can cause availability issues if not planned out correctly. Now, some IPS devices have failure technologies built-in to either fail-open or fail-closed in the case of a device failing. Fail-open means that if the IPS device fails, it will continue to pass traffic, and will just not inspect it. Fail-closed means that if the IPS is not able to inspect traffic, no traffic will pass. This decision will need to be made based on your security policy.

The goal here is to implement these technologies smartly, but still implement. Remember, Defense in depth is essential is today's world.

Structure

The thesis is structured in 4 chapters.

The first chapter deals with the creation and configuration of the network that will be used for this project and the proper illustration and configuration of the two network cards that will connect the hosts to the internet through the server.

Then we will illustrate the study of various applications, such as DHCP, DNS and Dynamic DNS, which are essential for the creation of such a system.

At the end we will install a web server on the machine, in this case Apache2 Web Server.

The second chapter is the most important part of this thesis, focusing on the aspects of network security in general and in more detail the network that was just implemented.

It will be studied the the behavior of a firewall in its various components.

We have chosen Netfilter with his relative tooI PTables, through which it is possible a total interaction with the Firewall.

For this task have chosen Snort, that is a software than can play both roles IDS and IPS:

in basic mode it will only detect the presence of threats, while in inline mode it will be able to block various intrusion attempts.

In chapter 3 we will explain examples and tests that we tried and the compiling of configuration files.

The fourth chapter includes all the tests and the attempts that have been carried out during the period of realization of this thesis and it we will show various programs during their execution.

In addition there is also a detailed bibliography which lists all the links to the web pages from which it we found the material and documentation. We also used two books for the study of each component of your project.

Chapter 1

1.1 Creation of a network

We will try to create a network fo this type :

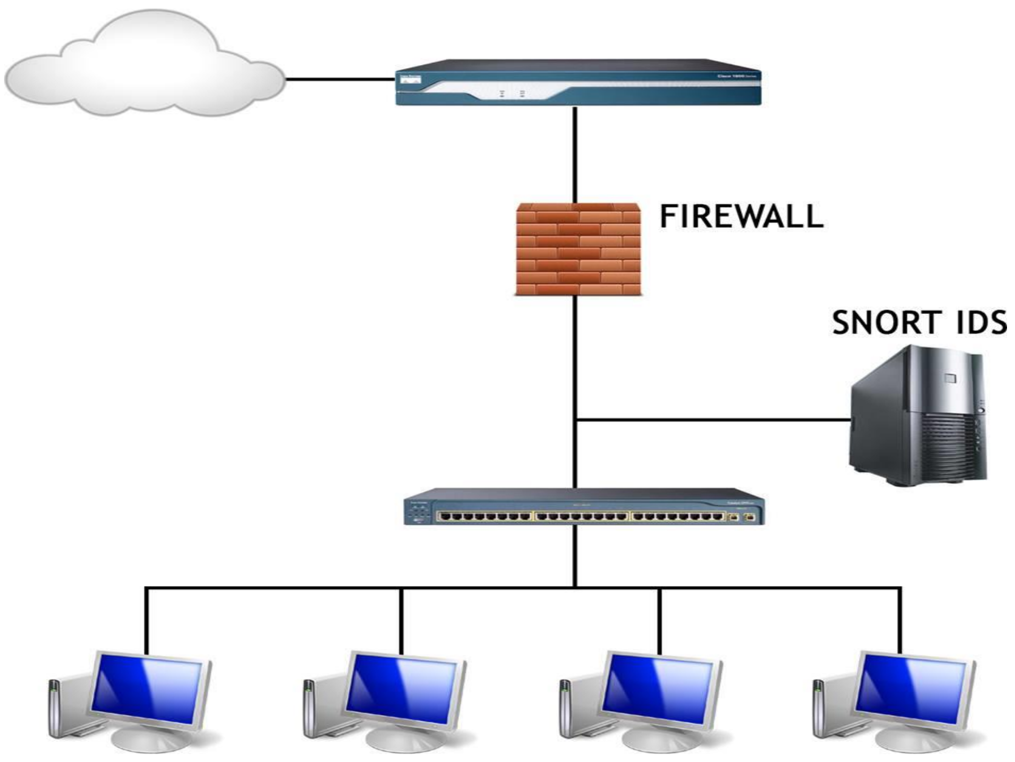


Figure 1: Our Type of Network

On the server we installed two network cards, eth0 and eth1. Eth0 will serve to connect to the external network(World Wide Web).

Eth1 will be connected to an internal subnet, formed from several other machines, and the computer with two network cards will become the server of reference for those connected to it through eth1.

Being then the server of the subnet it’s necessary, that this machine, provides the assignment of IP addresses, with which we can identify each computer present in the subnet.

This assignment can be done in two different ways, the first is static ip configuration, meaning that you can assign each machine a fixed IP address, that is, whenever the macchines make access to the network, will always be identified with the same IP.

In case you decide to assign addresses to computers on a network in a dynamic manner, it is necessary to have a DHCP server. For this to work you need to install an application that allows the server to complete this task.

in case you decide to assign addresses to computers on a network in a dynamic manner, it is necessary to have a DHCP server. For this to work you need to install an application that allows sull'Server all'Server to complete this task.

1.2 Operative System(OS) Installation

Before proceeding with the installation of any application we should chose what OS(Operative System) use.

In our case we chose ubuntu 12.04 Server.

Ubuntu is a Debian-based Linux operating system, with Unity as its default desktop environment. It is based on free software and named after the Southern African philosophy of *ubuntu* (literally, "human-ness"), which often is translated as "humanity towards others" or "the belief in a universal bond of sharing that connects all humanity".

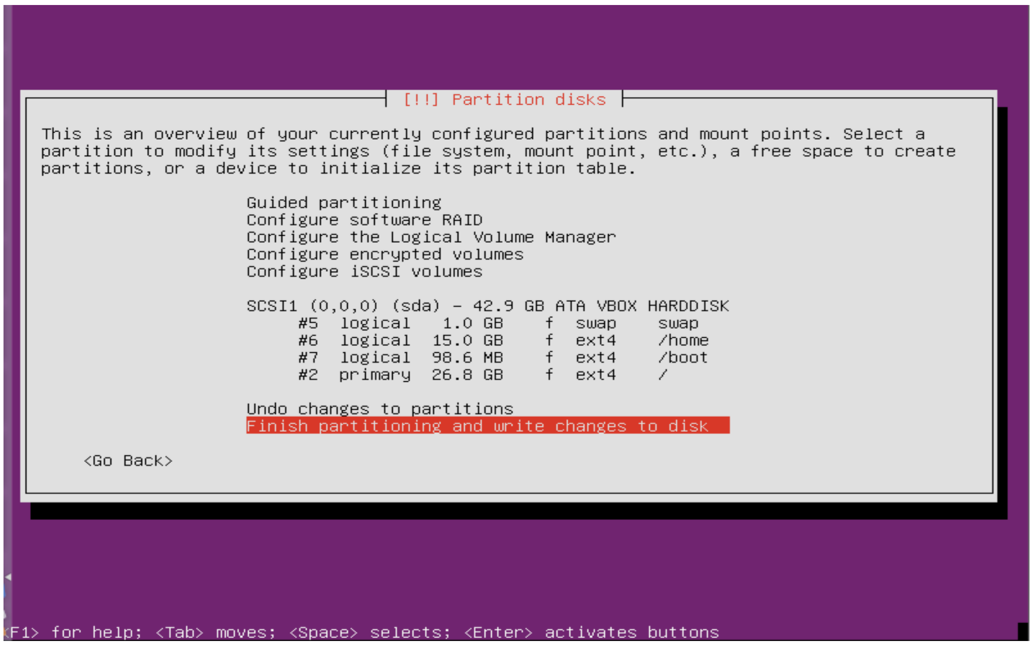
CentOS is available at http://www.ubuntu.com// and can be easily downloaded from the bottom of the main page.There are in fact, for each version of the operating system, various types, dedicated to different processor architectures, 32 or 64 bits, dedicated in particular to the server, or live distributions, that is bootable without the obligation to install, but directly from CD .

Once decided which of the different versions of 'OS responds to our needs, we download the image files from a server that is available on the website, and burn them onto optical media, then we can begin the installation process.

First, the user will be prompted for various information about the configuration of the system , such as the language in which it will be installed, keyboard layout, the arrangement of the characters on the keys, and finally the time zone for the country where you are located, so you can adjust so the system clock.

The next step concerns the definition of the space devoted to the partition where te system will be installed.

The file system to be adopted, having to do with a Linux system, will be the EXT4, remembering to reserve a bit of space for the partition dedicated to swap, the part of the hard disk used by the system for data-support once exhausted the RAM, more or less as is the case with the Paging File on Windows platforms, with the difference that Linux is used for this function a real hard drive partition.



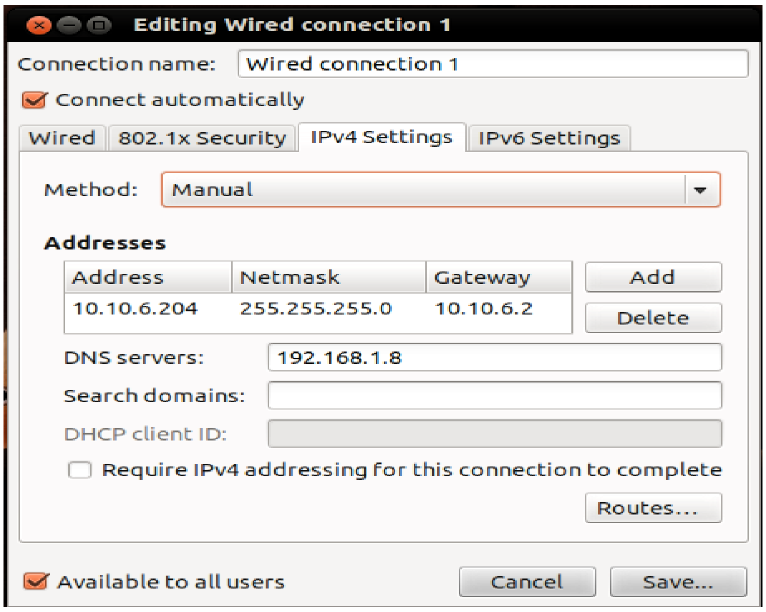
```
[!!] Partition disks

This is an overview of your currently configured partitions and mount points. Select a
partition to modify its settings (file system, mount point, etc.), a free space to create
partitions, or a device to initialize its partition table.

                    Guided partitioning
                    Configure software RAID
                    Configure the Logical Volume Manager
                    Configure encrypted volumes
                    Configure iSCSI volumes

                    SCSI1 (0,0,0) (sda) - 42.9 GB ATA VBOX HARDDISK
                         #5  logical    1.0 GB    f  swap    swap
                         #6  logical   15.0 GB    f  ext4    /home
                         #7  logical   98.6 MB    f  ext4    /boot
                         #2  primary   26.8 GB    f  ext4    /

                    Undo changes to partitions
                    Finish partitioning and write changes to disk

    <Go Back>

<F1> for help; <Tab> moves; <Space> selects; <Enter> activates buttons
```

Figure 2: Installing OS(Ubuntu)

Now we have to configure a key part for the realization of the security system: The Network Cards.

We ca do this by going to network manager and configure them in properly.

Here is a configuration example:



Editing Wired connection 1

Connection name: Wired connection 1

☑ Connect automatically

Method: Manual

Addresses

Address	Netmask	Gateway
10.10.6.204	255.255.255.0	10.10.6.2

DNS servers: 192.168.1.8

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

☑ Available to all users

Figure 3: Netword Card Configuration

But before we can access to the GUI(Graphical User Interface whe have to installi it with the command :

```
sudo apt-get install ubuntu-desktop
```

and then

```
sudo reboot
```

In the appropriate fields must be defined IP addresses, the eth0 will be facing the outside will have as its address:

193.205.92.119

The address will be dymanic, since the computer science department can not support it.

For the card eth1 should be assigned a fixed address such as:

192.168.1.1

1.3 Organization of the Network

The diagram of the system that we want to accomplish is the following:

The server through the network card eth1 is connected to the subnet through a switch, which is a network device that selectively forwards towards the exit door received packets, from the external network.

The server will then have the task of filtering traffic to and from the subnet, playing the role of gateways and firewalls.

The machines in the internal network do not require any configuration, but unlike the server computer, which having dual network device must have two different configurations, one for each card.

For the eth1 card will be sufficient to set the IP address to 192.168.1.1 and Subnet Mask to 255.255.255.0. Note that in this case the IP address is static, but it is to say that this will only happen for the server, as the machines on the subnet will be subject to dynamic assignment, via DHCP, given by the server.

1.4 DHCP

DHCP, or Dynamic Host Configuration Protocol, is a protocol that allows network devices to receive IP configuration necessary to operate on a network.

In a network based on the IP protocol, each computer needs the IP address, chosen in such a way that belong to the subnet to which it is connected and that it is unique, that there are not other computers that are already using that address.

This system removes the significant burden on network administrators to assign addresses manually, especially when it comes to large networks. There is also to say that with the rise of the machines on the Internet IPv4 fixed addresses have begun to dwindle, thus decreasing the chance that you might have un'indirizzo fixed.

The DHCP protocol consists of several components:

- The DHCP client is a computer that needs to get a valid IP address for the subnet to which it is attached, and is also the program that deals with request IP address and configure it.

- The DHCP server is the computer that assigns IP addresses, and it is also the process that performs this function. Sometimes this function is performed by a router.

- The DHCP relay is a computer that is responsible for forwarding requests to a DHCP server, if it is not on the same subnet, this component is required only if a DHCP server to provide multiple subnets. This function is often implemented in a router.

DHCP uses UDP and ports used are 67 and 68 respectively for servers and clients.

When a computer needs to obtain the address of the DHCP client initiates the process. It sends a package called DHCP DISCOVER broadcast, with a source IP address set as the

Convention to 0.0.0.0 and 255.255.255.255 as the destination. The packet is received by all hosts on the network and in particular from the DHCP server that can respond or not with a DHCPOFFER packet, which offers the client an IP address and other parameters.

This package is aimed directly at the datalink client, so it can only be sent from a server that is on the same broadcast domain.

If the domain there are one or more DHCP relay, they forward the packet to their server reference, which can respond to the client always through a relay. The relay agent tells the server its own IP address on the subnet from which it received the DHCP DISCOVER packet, allowing the server to figure out what subnet the request arrived, and then offer a right address for the subnet.

A DHCP server that is serving different IP subnets must be configured to know the parameters of each.

The client waits for a certain time to receive one or more offers, then select one, and sends a packet of DHC PREQUEST broadcast, pointing inside the package, with the field "server identifier", which reaches all DHCP servers on the network.

The server has been selected confirms the assignment of the address with a pack of DHC PACK, while other servers are informed that their bid was not chosen by the client, and that there is another subnet on the DHCP server.

At this point, the client is authorized to use the address received for a limited time, said lease time. Prior to the expiration of this time, it must renew groped by sending a new package DHC PREQUEST to the server, which will respond with a DHC PACK if he wants to extend the assignment of the IP address. If the client fails to renew its address, will come back to the initial state trying to take it in another attribute.

In this system, the application of choice for this task is isc-dhcp-server. Available from the repositories later, and they can be easily installed through the terminal command :

```
sudo aptitude install isc-dhcp-server  or
```

```
sudo apt-get install isc-dhcp-server
```

dhcp-client s already installed becouse without it we wouldn't have internet connection from the server to the world wide web.

The server will go then specially configured by editing the dhcpd.conf file, which is located in the directory ***/ etc / dhcp/dhcpd.conf*** or we can do it through Webmin(GUI).

Below is an example configuration DHCP file for the declaration of a subnet:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range                    192.168.1.100 192.168.1.200;

    option routers                          192.168.1.1;


    option subnet-mask                    255.255.255.0;


    option broadcast-address              192.168.1.255;

    option domain-name                     "google.com";

    option domain-name-servers              8.8.8.8;
}
```

Figure 4: Dhcp.conf file

1.5 DNS(Domain Name System)

The DNS or Domain Name System is a service used to resolve host names into IP addresses and vice versa. The name also denotes the protocol that governs the operation of the service, the programs that implement it, the servers on which they run, the set of these servers that cooperate to provide the service

The possibility to assign a textual name easy to store in a server greatly enhances the use of the service, as human beings is much easier to remember textual names rather than numbers.

Reverse resolution is useful for identifying the identity of the host, or to read the result of a traceroute.

To use the service, you must configure each client one or more of the root server.

The DNS uses UDP transport protocol and port 53 to meet the demands for a resolution from the host. DNS servers perform zone transfers using the TCP transport protocol at port 53.

The client side of the DNS is normally implemented by system libraries, which often integrate with other services for a resolution, so that a user can use a symbolic name in an application and get his resolution into an IP address without worrying about which instrument was used to obtain the resolution.

1.5.1 Dynamic-DNS

The term dynamic DNS, or DDNS, is a set of technologies that allow you to automatically fill in a DNS zone addresses of computers that do not get a static address, usually this is done through DHCP or PPP.

In a local network, this feature can be used directly by the client, or can be configured using the appropriate applications.

DDNS is also used by commercial services to allow users to dial-up (modem, ADSL) to register a name corresponding to the address that is assigned to them from time to time by their providers. In this way, a host with a dynamic IP address is always reachable. There DDNS client both in the form of applications that within router.There are DDNS client both in the form of applications that within router.

The dynamic DNS service is constituted by a series of dynamic client, from one or more servers and a communication protocol between the two parties. When a dynamic client obtains an IP address,it contacts one of the servers and informs him of his current IP address. The server then enters a DNS record that points to the new address of the customer. In this way, the other hosts are able to obtain the current IP address of the client using the normal dynamic DNS service, and therefore without being aware of the fact that the host which contact has a registered address dynamically.

1.6 Proxy

The proxy is a device that sits between a client and server, forwarding the requests and responses of both. The client connects to the proxy that cater to contact the server to accommodate client requests. Even vice versa will receive the server's response and will forward it to the client. Unlike a bridge or router, the proxy works at the application layer managing a limited number of application protocols.

Un proxy ha diversi scopi:

- keeping equipment in anonymity
- speed up access to resources (via caching)
- Filter the conten

A proxy server that receives requests and forwards them unaltered is usually called gateway or sometimes tunneling proxy.

1.6.1 Web proxy

A proxy that is focused on the www traffic is called "web proxy". The most common use of web proxy is to be used as a web cache.

Most proxy programs, such as Squid, provides the possibility to deny access to certain URLs contained in a blacklist, thus providing content filtering. This fact is usually used in a corporate environment, though with the increasing use of Linux, this function is no longer limited to large businesses but is also used in small businesses and homes.

1.6.2 Content-filtering web proxy

A web proxy server with content-filtering provides administrative control over the content.

This type of content filtering is used in commercial and non comercial areas (especially in schools) to ensure that Internet use is consistent with the policy of acceptable use of the network.

Here are some of the most common methods used for content filtering :

- URL or DNS blacklists
- URL filtering regex
- MIME filtering
- Content filtering with keywords

It can cooperate with anti-virus software to protect against viruses and malware by scanning incoming traffic, in real time.

1.6.3 Anonymize a connection through a proxy

An anonimous proxy server usually attempts to anonymize a connection. This is normal easy to circumvent by network administrators, and therefore is unusable.

There are different types of anonymizer. One of the most common variations is the open proxy, and because it is difficult to find, it is useful for those who seek anonymity online.

Using a proxy server to achieve anonymity, however, involves risks. In fact all the data that are sent pass through the proxy, and most of the times is in non-encrypted form. It is therefore a possible risk for the fact that the server can record everything that has been sent, including login details and passwords in clear text.

1.7 Installing the web server

Web servers are programs that allow the distribution of HTML pages on the Internet.

They are listening for requests for access to a website, processes it, and returns the data as a response. Such information, which contains all the necessary elements for viewing a web page consisting of text and images, are analyzed by the browser in the best way possible then presented to the user.

The web server communicate with the browser installed on the personal computer of the"client", ie the machine from which you are accessing through the http protocol.

It standardizes the process of sending and receiving data so that any client can communicate easily with any type of web server, without compatibility problems.

The operation of a web server, in the simplest case, it is the transmission of HTML pages, so this how the mechanism works:

- The browser requests an HTML page to the server
- The server retrieves the HTML page and sends it to the browser
- The browser requests other resources contained in the HTML page
- The server provides these resources to the browser that displays the page.

The ability of a server may, however, be increased by the use of server-side applications, that is, programs that run directly on the server.

1.7.1 Apache

Apache HTTP Server, or more commonly is the name given to the most widely used web server platform, is able to operate on Linux , UNIX and Windows operating systems

It's a software that performs the functions of transport information, and internetwork connections. It has the advantage of offering control functions for security as those who fulfilled by the proxy.

It is the most widely used web server in the world, born to run process as a "stand alone" ie without requiring the support of other applications or other software elements.

Since it is free it can be downloaded from the site, httpd.apache.org, as well as the application, available for both the Linux version and Windows platforms, including the source code of the program.

Apache is composed of as daemon on Unix and as service under Microsoft

The installation is very simple as we only have to open a terminal and type the command:

```
sudo apt-get install apache2
```

If your computer is connected to a network, in seconds the required packages will be downloaded and installed immediately.

The compilation of sources on the other hand is completed by performing a series of steps:

```
./configure && make && make install
```

Apache2 is configured by placing directives in plain text configuration files. These directives are separated between the following files and directories:

- apache2.conf: the main Apache2 configuration file. Contains settings that are global to Apache2.
- conf.d: contains configuration files which apply globally to Apache2. Other packages that use Apache2 to serve content may add files, or symlinks, to this directory.
- httpd.conf: historically the main Apache2 configuration file, named after the httpd daemon. The file can be used for user specific configuration options that globally effect Apache2
- mods-enabled: holds symlinks to the files in /etc/apache2/mods-available. When a module configuration file is symlinked it will be enabled the next time apache2 is restarted
- ports.conf: houses the directives that determine which TCP ports Apache2 is listening on.
- sites-available: this directory has configuration files for Apache2 Virtual Hosts. Virtual Hosts allow Apache2 to be configured for multiple sites that have separate configurations.
- sites-enabled: like mods-enabled, sites-enabled contains symlinks to the /etc/apache2/sites-available directory. Similarly when a configuration file in sites-available is symlinked, the site configured by it will be active once Apache2 is restarted.

These ./configure option allows you to enable specific features or to define specific parameters :

--prefix	- Sets the base directory in which to install all files
--enable-layout=layout	-Sets a default layout appropriate for the actual system
--with-ssl=/path/lib/ssl	-Enable SSL support (mod_ssl required)

Chapter 2

2.1 Network Security

From the moment you interconnect multiple computers between them lies the problem of network security.
A network, in fact, is subject to various types of vulnerabilities, which can be exploited by a third party to intercept the data traffic.

In the early years of the existence of networks, they were mainly used by university researchers to send and receive email, and businesses to share printers among them.
Today millions of people use networks to shop, to do transactions,or even to share classified data and since everything is on the www the major problem is to secure this data so intruders can’t take possess of them in an non-legal way.

The problems of security of networks can be roughly divided into four interconnected areas:

- Privacy
- Authentication
- Non-repudiation
- integrity Check

The secrecy, also called confidentiality, is responsible for keeping the information out of the reach of unauthorized users. This represents the idea of security that people usually have.

Authentication is concerned with establishing the identity of the person with whom you are communicating with before revealing sensitive information or enter into commercial transactions, etc..

Regarding digital security, the cryptological meaning and application of non-repudiation shifts to mean:

- A service that provides proof of the integrity and origin of data.
- An authentication that can be asserted to be genuine with high assurance

The integrity check of the data or objects is carried out to verify that there has been tampering, or alteration or even destruction of part or all of them.

For systems on networks, especially on the Internet, the threats to be resisted are many and of different magnitudes:

- Viruses, worms and other automated agents, which spread themselves in out of date systems, without the will and human intervention, and can cause damage to the entity variables.
- Large-Scale scans searching for vulnerable systems by exploiting security gaps generally known. These scans can be made from various types of attackers, such as script kiddies or crackers. The latter are really dangerous, because they have a knowledge of highly sophisticated attack techniques.
- Targeted attacks determined by the penetration of one or more hosts of a specific entity for various modes. They are insidious and more dangerous, because if carried out by capable and determined cracker can compromise also well protected systems

The security policies that need to be used have different levels of complexity. While it is relatively easy to defend against the first two types of threats, setting up a structure that exposes only the services that are strictly necessary and updating all the time the software that runs these services, it becomes much more difficult to achieve an extreme safe system, able to withstand the attacks more savvy or via internet by indirect means.

There are no certainties in the field of information security, a 100% safe network system does not exist, a software that seems secure today, tomorrow can be found exploitable gaps for intruders. A network must be protected as administrable and who administers or uses special privileges, can be a threat by itself if it is not good enough. As the security issues are so large and have different factors, one can define some minimum guidelines for designing a network as possible protected from those which, in fact, are the major threats: unknown intrusion from the Internet. Remote access to a system can take place almost exclusively through an open TCP or UDP port on a host on the network, so at the level of networking control and attention should be placed on which ports to allow access and how.

A network with a secure base design requires the application of some simple basic rules that need to be adapted to different contexts:

- Close the ports of unused public IP
  Any attempt of intrusion can only be done remotely through any open ports on a host. The first, basic, convenient and useful procedure to be adopted to start protecting your network is to not leave gaps useless to potential intruders. The closing of the ports can be made both at the level of individual hosts, removing all services that are not needed, both at the perimeter firewall, with strict packet filtering. The two solutions are not alternatives, indeed can safely co-exist on the host removing unnecessary services, on the firewall, you can manage which source IP addresses can access certain services.
- Centralize the control points of the traffic
  Typically a network increases its complexity with its natural growth, but if it is not structured from the beginning so farsighted, threatens to become a tangle of routers and firewalls on which action should be taken in case of problems or "door openings." Centralizing the routing and firewalling points helps to more easily implement changes to existing configurations, and diagnose network problems, as well as reducing the infrastructure point of failure. The perimeter firewall should, by default, block all types of packet from the outside and provide specific rules for each server in production. These rules can be of two types:

  a) Rules that open a specific port on a given host to the entire Internet. These are the rules necessary for public services in production (web server, mail server, DNS, etc..). And it is a good thing that they "open" only the ports strictly necessary . In the Firewalls that control filtering rules in a sequential manner (chains of iptables, Cisco IOS ACLs or PIX ...) it is always better to first put the rules concerning traffic flows , to lessen the impact on resources by avoiding unnecessary testing of less used filtering rules.

b) Rules which give access to certain ports and IP's from specific IP sources. We may need them to gain FTP access to the data feed provider or allow VPN access from a particular location or to access to a database from a remote server. These typically tend to make more complicated the configuration of a firewall, since they have to adapt to specific source IP's, but since they are necessary they it is neccessary to centralize.
In applying such rules, which tend to weigh down their numbers to the firewall, it is recommended that common sense and tendency to grouping: if there are multiple IP to "open" in the same subnet, you can consider opening the entire subnet, if there are more ports to open (especially doors that actually allow access to the system, such as telnet and ssh) may make sense to open up access to the entire IP address of the destination host, without specifying individual ports

- Check all access points
  In terms of computer security, the weakest link, which means that less controlled and secure one, reduces the security of the entire infrastructure. For this reason it is essential to identify and ponder each point of access to our systems remotely. This includes routers, customers , suppliers and machines that allow access via modem to the network that is to be protected.

- Multilayer design in the case of complex networks

  If the network that is to be managed includes dozens of host of different uses (front-end production machines backend machines as the database or monitoring systems , internships and pre-production machines development , client systems analysts and developers , etc.) it must be considered a subdivision of our network in multiple layers, trying to protect the most interior and delicate one (the backend) and limiting as much as it can the access to these machines from other machines that are more exposed (for example, public front-end servers, development servers or clients) . In complex structures, if possible, front-end machines should not have access in writing to the data : a web server for example, should not be able to write to the directory that contains its web pages , generated by a Content Management System on the backend. With simpler networks is still a good idea to try to limit the communication possibilities of a machine with any other machine in the network , so as to limit the potential damage on all systems from a break in one of the hosts .

2.2 Diversity in defense

Having a diversity in our difense system is an important aspect regarding the configuration of a system discreetly secure. It is not all we need having a good password, firewall or IDS/IP, there is always room for improvement. The system has to be configured in a way that when crackers attempt to enter in that system, they sure will have difficulties to do their tricks. But the tougher and complicated is the defence of the system, the more difficult with will be the managing part from the SYSADMIN.

The three fundamental principles of creating and organizing an standard secure network are:

- Minimize the number of open ports exposed to the internet
- Being in possess of updated software which listen on those open ports
- Configure in a correct way the public services

2.3 Firewall

The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation (DEC) developed filter systems known as packet filter firewalls. This fairly basic system was the first generation of what would become a highly evolved and technical internet security feature.

A firewall is a system designed to prevent unauthorized access to or from a private network. You can implement a firewall in either hardware or software form, or a combination of both. Firewalls prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet (i.e., the local network to which you are connected) must pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Figure 5: Firewall

A firewall can be implemented(configured) in a normal computer that is provided with at least two network adapters, it can be a function included in a router or a specialized hardware. There are also so-called personal firewalls, which are programs installed on normal computers, which filter only packets that enter and leave the computer, in that case you use one network card only.

We have several types of firewalls :

- **Packet filtering:** The system examines each packet entering or leaving the network and accepts or rejects it based on user-defined rules. Packet filtering is fairly effective and transparent to users, but it is difficult to configure. In addition, it is susceptible to IP spoofing.
- **Circuit-level gateway implementation:** This process applies security mechanisms when a TCP or UDP connection is established. Once the connection has been made, packets can flow between the hosts without further checking.
- **Acting as a proxy server:** A proxy server is a type of gateway that hides the true network address of the computer(s) connecting through it. A proxy server connects to the Internet, makes the requests for pages, connections to servers, etc., and receives the data on behalf of the computer(s) behind it. The firewall capabilities lie in the fact that a proxy can be configured to allow only certain types of traffic to pass (e.g., HTTP files, or web pages). A proxy server has the potential drawback of slowing network performance, since it has to actively analyze and manipulate traffic passing through it.
- **Web application firewall:** A web application firewall is a hardware appliance, server plug-in, or some other software filter that applies a set of rules to a HTTP conversation. Such rules are generally customized to the application so that many attacks can be identified and blocked.

The firewall acts on the packages in transit to and from the inner zone being able to perform these operations on them:

- Control
- Editing
- Monitoring

It can do those three operations due to its ability to "open" the IP packet and to read the information on its header, and in some cases also to verifies the contents of the package.

The syntax of the configuration of a firewall in many cases is based on a mechanism of access control list (ACL), which can be static (then changed only by explicit configuration) or dynamic (ie, which can vary depending on the internal state of the system, such as in the Port knocking).

A function that is often associated with the firewall is NAT (network address translation), which may contribute to disable access to computers on the internal network.

Sometimes a firewall is also associated with the function of intrusion detection (IDS), a heuristic-based system that analyzes traffic and tries to recognize the possible attacks on the security of the network, and can also trigger automatic reactions by the firewall (Intrusion Prevention System) like in our case.

2.3.1 Iptables Firewall

Netfilter is a component of the kernel of the Linux operating system, which allows the interception and manipulation of packages.

Iptables is the program that allows system administrators to configure netfilter, defining the rules for the use of network filters and NAT redirection. Often the term iptables refers to the entire infrastructure, including netfilter. Netfilter is used both in computers that are used as hosts to achieve real routers based on Linux.

The system is based on netfilter rules grouped into chains, and every chain is grouped in tables. Each table defines a different type of operations you can perform on packets, each chain defines how packets are processed at different stages of their development.

The chains are a form of access control list: each rule consists of two parts: the specification of the characteristics that a packet must have in order that the same rule is applied and a goal or target, which indicates what to do when the package meets the specifications given.

There are three predefined tables, each of which contains built-in chains. There is also the possibility to create other tables. Initially, all chains are empty and have a policy that allows all packets to pass through without being blocked or altered in any way, but they can be edited if needed.

Predefined Tables :

a) Filtering table: Is responsible for packet filtering, that allows packets blocking or allowing . Each packet passes through the filter table. It contains the following built-in chains:

- INPUT chain: all packets destined to the system go through this chain.
- Chain OUTPUT: all packages created by the system go through this chain.
- FORWARD chain: all packages that have as their final destination another system and that have not been generated by the system itself, ie all packets that are routed from the system merely pass through this chain.

b) NAT table : This table is responsible for setting the rules for changing the addresses and ports of the packets. The nat table contains the following built-in chains:

- PREROUTING chain: the chain pass through incoming packets, the transition occurs before the local routing table is consulted to perform the routing. It is used for the destination NAT or DNAT.
- POSTROUTING chain: outgoing packets pass through this chain after the local routing table is consulted. Used for NAT on the source or SNAT.

- Chain OUTPUT: allows a limited DNAT on locally generated packets.

c) Mangle table : This table is responsible for changingthe options of the packages, such as the one that determines the quality of the service. All packets pass through this table.

   It contains all the built-in chains:

   - PREROUTING chain: examines all packets that enter the system. This process takes place before the routing decides whether the packet should be forwarded (FORWARD chain) or whether it is intended for the system. It is used to manipulate the header of the packet (INPUT chain).
   - INPUT chain: all packets destined to the system go through this chain.
   - FORWARD chain: all packets that are routed from the system, but the system is neither the initial source nor it final destination, pass through this chain.
   - Chain OUTPUT: all packages created by the system go through this chain.
   - POSTROUTING: all packets that leave the system, both in OUTPUT and FORWARD, pass through this chain.

Rules structure :

- The purpose of a rule is the action to take if a packet meets the rule, and is specified with the following option:

  -j target   ,   --jump target

  The target can be a chain defined by the client, ACCEPT, DROP,

  QUEUE, RETURN, REJECT,LOG or

  MASQUERADE(we used this for the assingment of dynamic IP’s for our hosts.)

2.3.2 Interface GUI-Firewall(Webmin)

Webmin is a web-based interface for system administration for Unix. Using any browser that supports tables and forms (and Java for the File Manager module), you can setup user accounts, Apache, DNS, file sharing and so on.

It consists of a simple web server, and a number of CGI programs which directly update system files like /etc/inetd.conf and /etc/passwd. The web server and all CGI programs are written in Perl version 5, and use no non-standard Perl modules.

Webmin is a web-based system configuration tool for Unix-like systems, although recent versions can also be installed and run on Windows. With it, it is possible to configure operating system internals, such as users, disk quotas, services or configuration files, as well as modify and control open source apps, such as the Apache HTTP Server, PHP or MySQL.

Webmin is largely based on Perl, running as its own process and web server. It defaults to TCP port 10000 for communicating(https://localhost:10000) , and can be configured to use SSL if OpenSSL is installed with additional required Perl Modules.

It is built around modules, which have an interface to the configuration files and the Webmin server. This makes it easy to add new functionality. Due to Webmin's modular design, it is possible for anyone who is interested to write plugins for desktop configuration.

Webmin also allows for controlling many machines through a single interface, or seamless login on other webmin hosts on the same subnet or LAN.

Besides all this beautiful staff that Webmin can do it gives us a possibility to configure Iptables(Firewall) graphically.

This is how we Webmin can be configured :

- Prior to installing Webmin we need to install a few pre-requisites. Type the following command below to install the required modules:

```
# sudo apt-get install libnet-ssleay-perl libauthen-pam-perl libio-pty-perl
apt-show-versions libapt-pkg-perl -y
```

- Now we can download Webmin with this command:

```
#sudo wget http://prdownloads.sourceforge.net/webadmin/webmin_1.630_all.deb
```

Note: Your URL may be different as new versions of the webmin application are released

- Now it is time to install the Webmin package. To do so, use the command below:

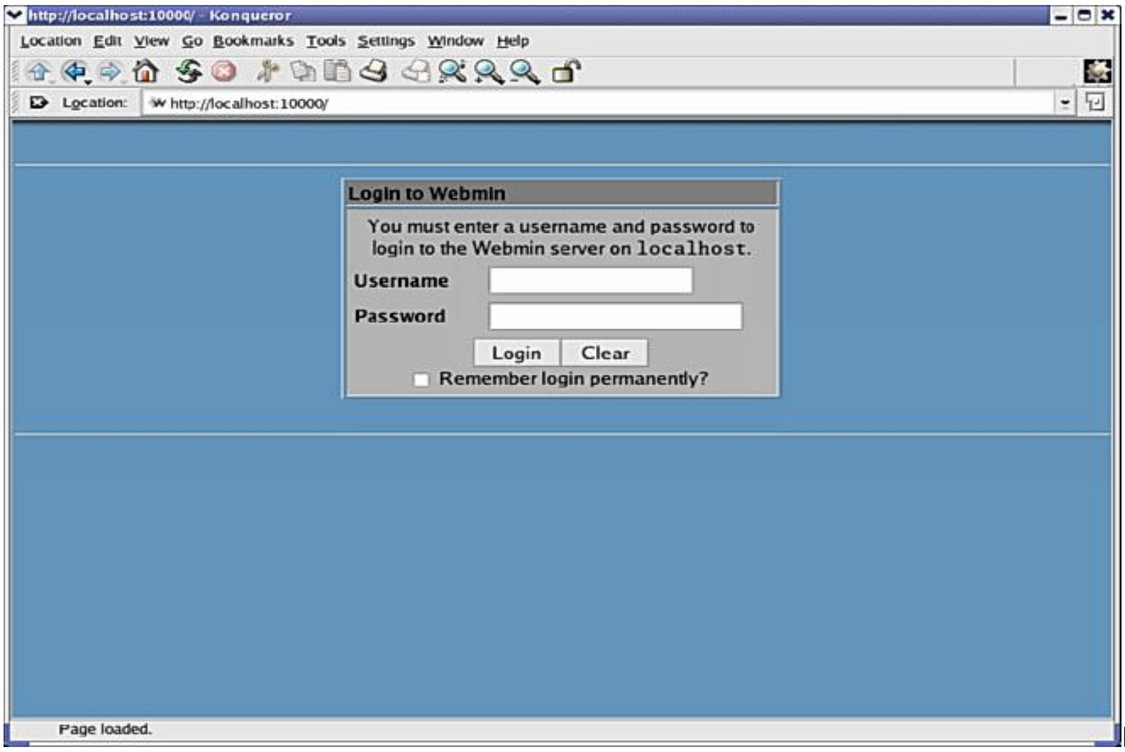
```
dpkg -i webmin_1.630_all.deb
```

- Once the installation has completed you should get the following message:

```
Webmin install complete. You can now login to https://localhost:10000/
as root with your root password, or as any user who can use sudo
to run commands as root.
```

- Then we can open a browser, type https://localhost:10000/ and that is it.

- Input username and password and we can access to Webmins GUI.


http://localhost:10000/ - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: http://localhost:10000/

Login to Webmin

You must enter a username and password to login to the Webmin server on localhost.

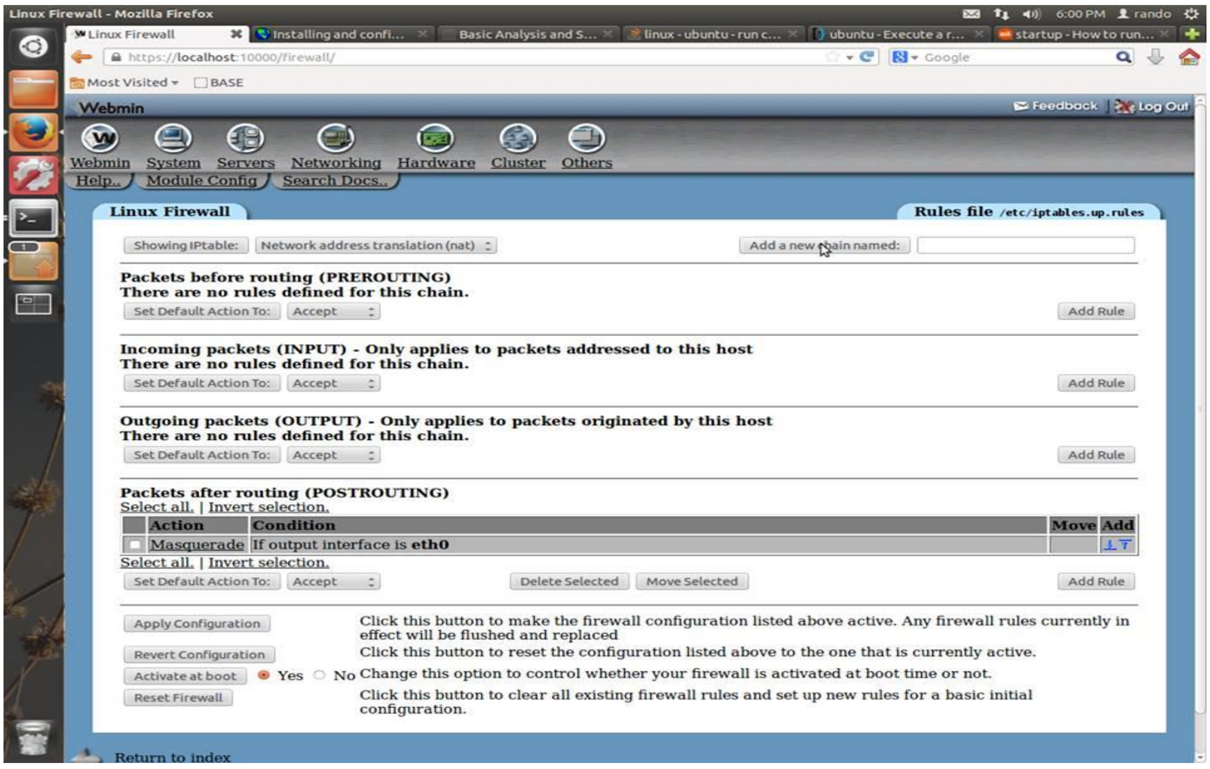
Username	
Password	

Login | Clear

☐ Remember login permanently?

Page loaded.

Figure 6: Login inteface(Webmin)



Linux Firewall Rules file /etc/iptables.up.rules

Showing IPtable: Network address translation (nat) | Add a new chain named:

Packets before routing (PREROUTING)
There are no rules defined for this chain.
Set Default Action To: Accept | Add Rule

Incoming packets (INPUT) - Only applies to packets addressed to this host
There are no rules defined for this chain.
Set Default Action To: Accept | Add Rule

Outgoing packets (OUTPUT) - Only applies to packets originated by this host
There are no rules defined for this chain.
Set Default Action To: Accept | Add Rule

Packets after routing (POSTROUTING)
Select all. | Invert selection.

	Action	Condition	Move	Add
☐	Masquerade	If output interface is **eth0**		

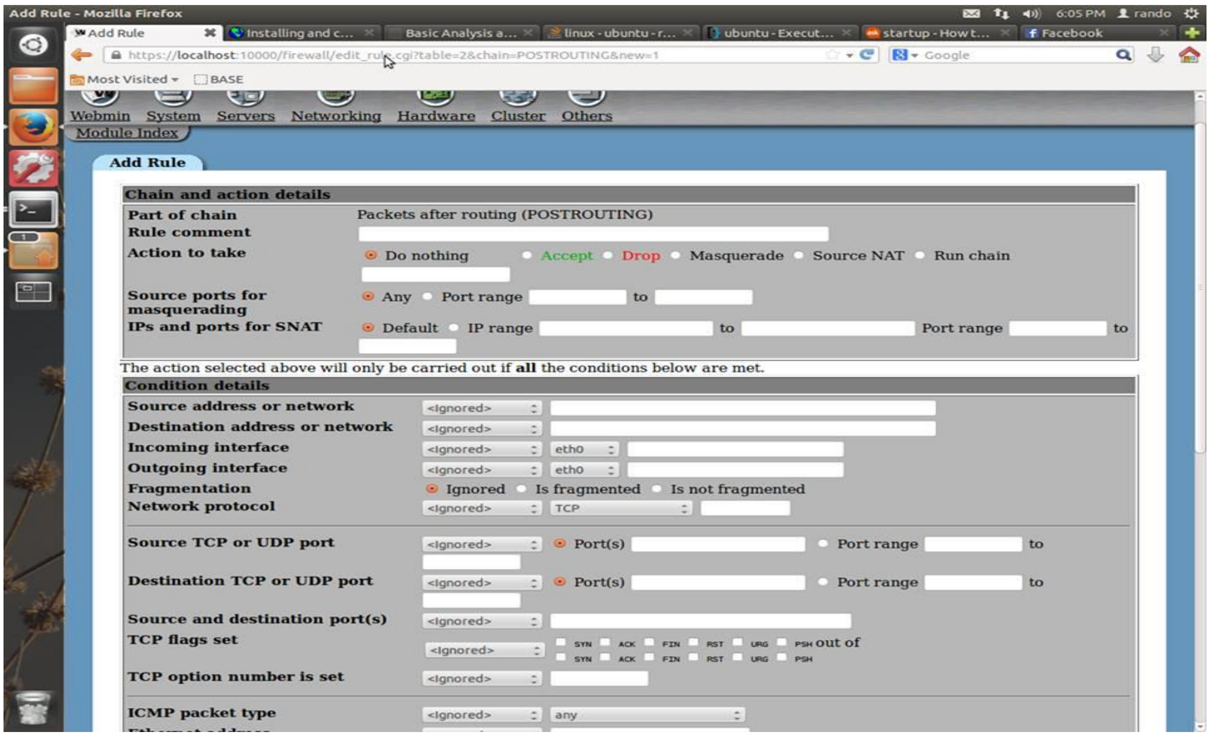
Select all. | Invert selection.
Set Default Action To: Accept | Delete Selected | Move Selected | Add Rule

Apply Configuration	Click this button to make the firewall configuration listed above active. Any firewall rules currently in effect will be flushed and replaced
Revert Configuration	Click this button to reset the configuration listed above to the one that is currently active.
Activate at boot [✓ Yes] No	Change this option to control whether your firewall is activated at boot time or not.
Reset Firewall	Click this button to clear all existing firewall rules and set up new rules for a basic initial configuration.

Return to index

Figure 7: Firewall GUI from Webmin

This is what Webmin looks after we provide valid username and password , now go to /networking/Linux Firewall and this is where the firewall can be configured.


Add Rule - Mozilla Firefox
https://localhost:10000/firewall/edit_rule.cgi?table=2&chain=POSTROUTING&new=1
Webmin
System
Servers
Networking
Hardware
Cluster
Others
Module Index

Add Rule

Chain and action details

Part of chain	Packets after routing (POSTROUTING)
Rule comment	
Action to take	[✓ Do nothing] Accept Drop Masquerade Source NAT Run chain
Source ports for masquerading	[✓ Any] Port range to
IPs and ports for SNAT	[✓ Default] IP range to Port range to

The action selected above will only be carried out if **all** the conditions below are met.

Condition details

Source address or network	<Ignored>
Destination address or network	<Ignored>
Incoming interface	<Ignored> eth0
Outgoing interface	<Ignored> eth0
Fragmentation	[✓ Ignored] Is fragmented Is not fragmented
Network protocol	<Ignored> TCP
Source TCP or UDP port	<Ignored> [✓ Port(s)] Port range to
Destination TCP or UDP port	<Ignored> [✓ Port(s)] Port range to
Source and destination port(s)	<Ignored>
TCP flags set	<Ignored> SYN ACK FIN RST URG PSH out of SYN ACK FIN RST URG PSH
TCP option number is set	<Ignored>
ICMP packet type	<Ignored> any

Figure 8: Firewall Rules in Webmin

Clicking add rule and will give us this interface, and the rule can be configured from zero as the user wants.

The rule will be saved in /etc/iptables.up.rules and if aply configuration is clicked than everything will be saved for the next time we use iptables.

It can be done even from the terminal though executing this command :

To save the configuration, you can use `iptables-save` and `iptables-restore.`

2.4 IDS/IPS(Snort)

Snort is an open source network intrusion prevention system (IPS) capable of performing real-time traffic analysis and packet-logging on IP networks. It can perform protocol analysis, content searching & matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts and more.

Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that uses a modular plug-in architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user-specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient. Snort has three primary uses. It can be used as:

- a straight packet sniffer like tcpdump,
- a packet logger (useful for network traffic debugging and so), or as
- a full-blown network intrusion prevention system.

The system needs a machine with two network cards, respectively Eth0 and Eth1, on which is installed a firewall. The device Eth0 will be facing outwards and then exposed to Internet traffic. The second network card will be attached to the inside through a HUB, Switch, or host(for testing) to an internal network, with a fixed address like 192.168.1.0/24. The machine in question will provide the whole subnet of IP addresses via DHCP, while also providing DNS and Dyn-DNS services. Snort will be installed in the same machine that we installed the firewall.

2.4.1 Installing Snort

- First of all we need to install all the fllowing packages :

```
# apt-get update && apt-get -y install apache2 apache2-doc autoconf automake
bison ca-certificates ethtool flex g++ gcc gcc-4.4 libapache2-modphp5
libcrypt-ssleay-perl libmysqlclient-dev libnet1 libnet1-dev libpcre3 libpcre3-dev
libphp-adodb libssl-dev libtool libwww-perl make mysqlclient
 mysql-common mysql-server ntp php5-cli php5-gd php5-mysql php-pear sendmail
```

- After that we need to install and configure three important components for snort:

- ✓ Libpcap
- ✓ Libdnet
- ✓ Daq

- ➢ At this point we can begin to install Snort(www.snort.org)

```
cd /usr/src && wget http://labs.snort.org/snort/2930/snort.2930.conf -O snort.conf
```
(snort configuration file)

```
wget http://www.snort.org/dl/snort-current/snort-2.9.3.tar.gz -O snort-2.9.3.tar.gz
```
(snort itself)

snort.conf must be configured according to the needs that the system has

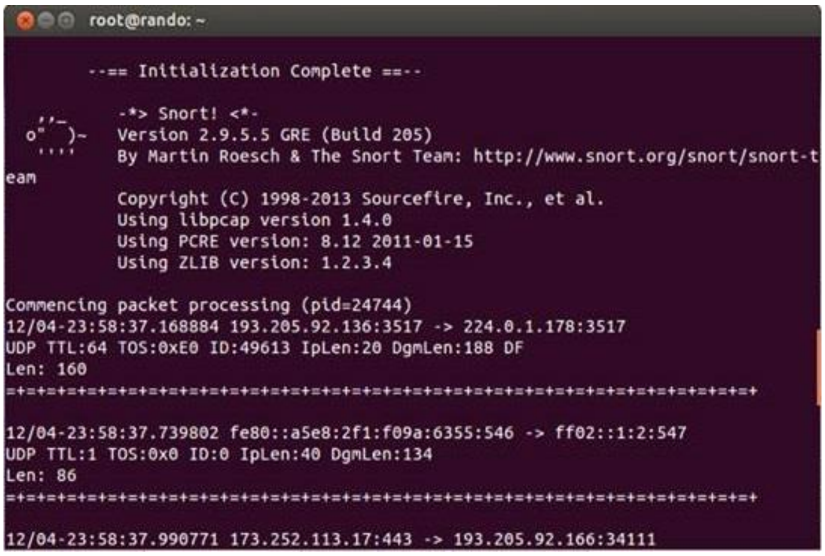
For more detailed information we will use this guide :

https://www.snort.org/assets/167/IDS_deb_snort_howto.pdf

After we finish installing snort, we should se if is installed correctly by typing :

```
#snort –v
```

And we should get a something like this :



```
root@rando: ~

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.5.5 GRE (Build 205)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t
eam
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.4.0
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.3.4

Commencing packet processing (pid=24744)
12/04-23:58:37.168884 193.205.92.136:3517 -> 224.0.1.178:3517
UDP TTL:64 TOS:0xE0 ID:49613 IpLen:20 DgmLen:188 DF
Len: 160
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-23:58:37.739802 fe80::a5e8:2f1:f09a:6355:546 -> ff02::1:2:547
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:134
Len: 86
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/04-23:58:37.990771 173.252.113.17:443 -> 193.205.92.166:34111
```

Figure 9: Snort version

2.4.2 Installing Barnyad2

Barnyard2 is an open source interpreter for Snort unified2 binary output files. Its primary use is allowing Snort to write to disk in an efficient manner and leaving the task of parsing binary data into various formats to a separate process that will not cause Snort to miss network traffic.Barnyard is an output system for Snort. Snort creates a special binary output format called unified. Barnyard reads this file, and then resends the data to a database backend. Unlike the database output plug-in, Barnyard is aware of a failure to send the alert to the database, and it stops sending alerts. It is also aware when the database can accept connections again and will start sending the alerts again."

Barnyard2 has 3 modes of operation:

- batch (or one-shot)
- continual
- continual w/ bookmark

In batch (or one-shot) mode, barnyard2 will process the explicitly specified file(s) and exit.

In continual mode, barnyard2 will start with a location to look and a specified file pattern and continue to process new data (and new spool files) as they appear.

Continual mode w/ bookmarking will also use a checkpoint file (or waldo file in the snort world) to track where it is. In the event the barnyard2 process ends while a waldo file is in use, barnyard2 will resume processing at the last entry as listed in the waldo file.

2.4.3 Configure MYSQL

To ensure that Snort has the ability to log the alerts is required a database on which to keep track of events that have occurred.

In the specific case will be adopted an open source database management system, called MySQL.

MySQL is a DBMS that consists of a client and a server, both available for Linux and Windows platforms.

The MySQL installation generally includes the use of different packages (or compilation of its components):

- mysql-server - The real server
- mysql-client - The command line client
- mysql-devel – Librerie

In terms of password management and access is recommended:

- Set a password for the root user.
- Set login / passwords for each application or website that works on a single database, so that this user can work only on the specific db.

- If it is needed to query the data, it is sufficient to use the SELECT permissions , but if you need to edit it, it should be enough to use the following permitions SELECT, UPDATE, INSERT, DELETE.
- The port on which MySQL listens (TCP 3306) should never be exposed to the Internet, it is sufficient that it is accessible from the host on which it runs the application that uses the DB server (such as a web server with php pages).

Now we will create the database that will be used by Snort:

```
# mysql -u root –p   #You will be prompted to enter the password you created during installation

mysql> create database snort;

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;

mysql> SET PASSWORD FOR snort@localhost=PASSWORD('mypassword'); # set user password
different from "root" password

mysql> use snort;

mysql> source /usr/src/create_mysql

mysql> show tables; # you should see the list of new tables you just imported.

mysql> exit
```

We have to add this line in the snort.conf file so we can use snort for the output database :

```
output database: log, mysql, user=snort password=<mypassword> dbname=snort host=localhost
```

These commands create a new user "snort" with permissions to create, insert data, select, delete, and update the database snort on all tables that comprise (. *), all carried out on the local machine.

2.4.4 Snort Rules(Oinkmaster)

Oinkmaster is a script that will help you update and manage your Snort rules. It is released under the BSD license and will work on most platforms that can run Perl scripts, e.g. Linux, *BSD, Windows, Mac OS X, Solaris, etc. Oinkmaster can be used to update and manage the VRT licensed rules, the community rules, the bleeding-snort rules and other third party rules, including your own local rules.

Oinkmaster's contrib directory contains several useful scripts related to rules management, like adding SIDs to rules that don't have any, creating SID maps (sid-msg.map), and so on.

You can mark certain rules as being "locally modified" to prevent them from being updated.

Oinkmaster can read any number of configuration files, either specified on the command line or by using 'include' statements, so you can use one global config and one sensor-specific config for finetuning etc.

Can be used in conjunction with other programs using Snort rules, like Prelude-NIDS.

To install oinkmaster we will follow these three steps :

1) Put oinkmaster.pl in some suitable directory, for example
/usr/local/bin/. Put oinkmaster.conf in /etc/ or /usr/local/etc/
(this is where Oinkmaster will search for it by default).
You may also want to copy the man page (oinkmaster.1) to
something like /usr/local/man/man1/.

2) Edit oinkmaster.conf that you copied in step 1). The defaults should
be fine for most users, although one thing you must change is
"url = ", which specifies the location of the rules archive.
The URL to use depends on which version of Snort you run and also what
type of rules you want to use. Some may require registration. See Q1

in the FAQ for more information.

3) Decide in which directory you want to put the new rules. If you
have Snort up and running already, you should use the directory where
you keep the rules files. It's a very good idea to create a backup of
it first. You must run Oinkmaster as a user that has read/write access
to your rules directory and all rules files in it. It should however
NOT be a privileged user such as root!
Never run Oinkmaster as root.

Our rules directory is /etc/snort/rules/, we can now update
those rules by running:

```
oinkmaster.pl -o /etc/snort/rules
```

And the rules should have been downloaded

2.4.5 Interface GUI-snort(BASE)

BASE is the Basic Analysis and Security Engine. It is based on the code from the Analysis Console for Intrusion Databases (ACID) project. This application provides a web front-end to query and analyze the alerts coming from a SNORT IDS system.

BASE is a web interface to perform analysis of intrusions that snort has detected on your network. It uses a user authentication and role-base system, so that you as the security admin can decide what and how much information each user can see. It also has a simple to use, web-based setup program for people not comfortable with editing files directly.

BASE is supported by a group of volunteers. They are available to answer any questions you may have or help you out in setting up your system. They are also skilled in intrusion detection systems and make use of that knowledge in the development of BASE.

This is how base will be installed :

```
# cd /tmp && wget http://sourceforge.net/projects/secureideas/files/latest/download
```
the directory and link where do BASE will be downoaded

```
# tar -xzvf base-1.4.5.tar.gz
```
unziping the file

```
# cp -r base-1.4.5/ /var/www/base
```
change name of the directory

```
# cd /var/www/base/
```
enter to the BASE directory

```
# cp base_conf.php.dist base_conf.php
```
copy files in directory

For the configuration, it must be edited the file base_conf.php.dist, rename it to base_conf.php, and in which it is needed to edit the following parameters:

```
#vi base_conf.php
```

```
# BASE_urlpath = '/base';
```

the field is emty so we insert base

```
#DBlib_path = '/usr/share/php/adodb';
```

inserting adodb directory

```
#alert_dbname = 'snort';
```
we will put here the name of the database that was created earlier

```
#alert_host = 'localhost';
```
this is the name of the machine where the database is located

```
#alert_port = '3306';
```
The port of the alerts, it can be leave empty if wanted

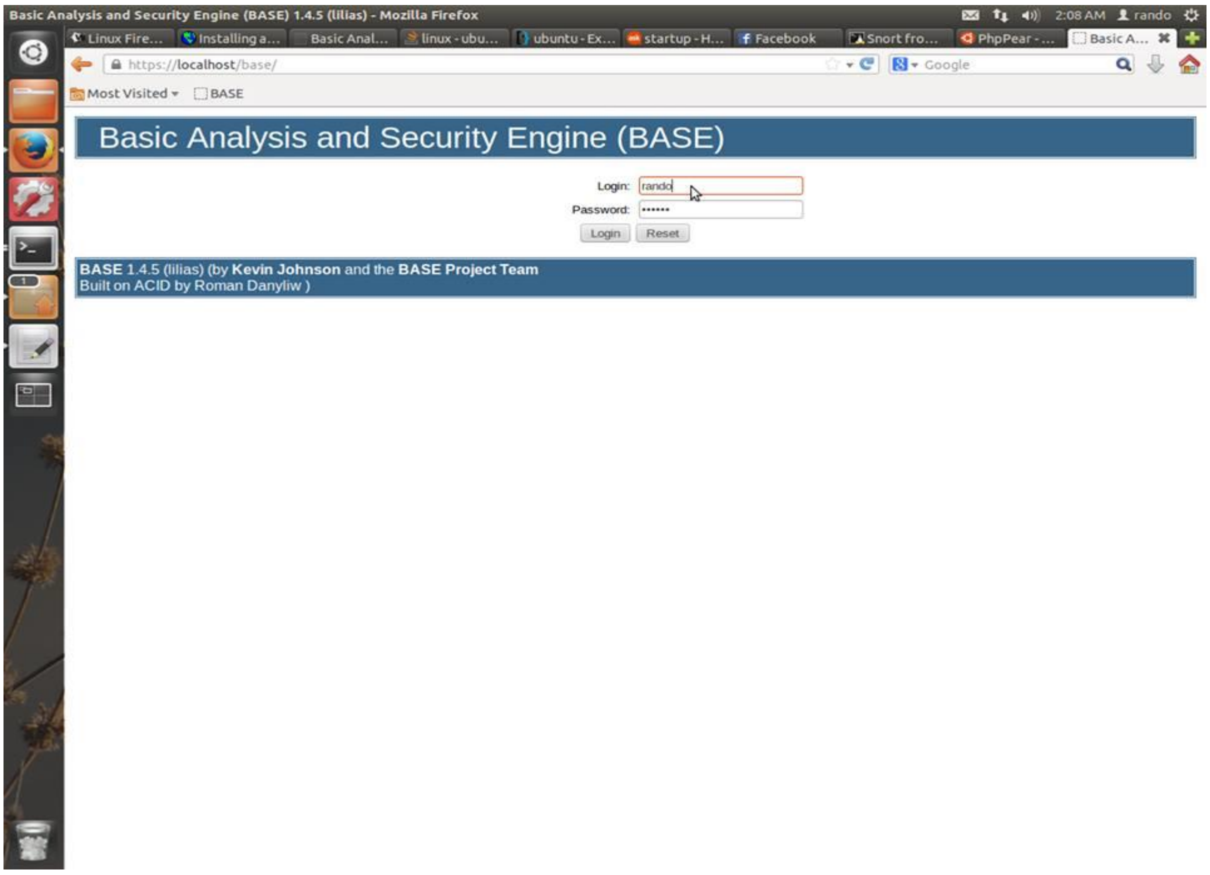
```
#alert_user = 'snort';
```
Must insert the database username

```
#alert_password = 'snortpassword';
```
Must insert the database password

After everything is ready we can open a browser and type

https://localhost/base

and the result must be like this :


Basic Analysis and Security Engine (BASE) 1.4.5 (lilias) - Mozilla Firefox
https://localhost/base/
Most Visited
BASE

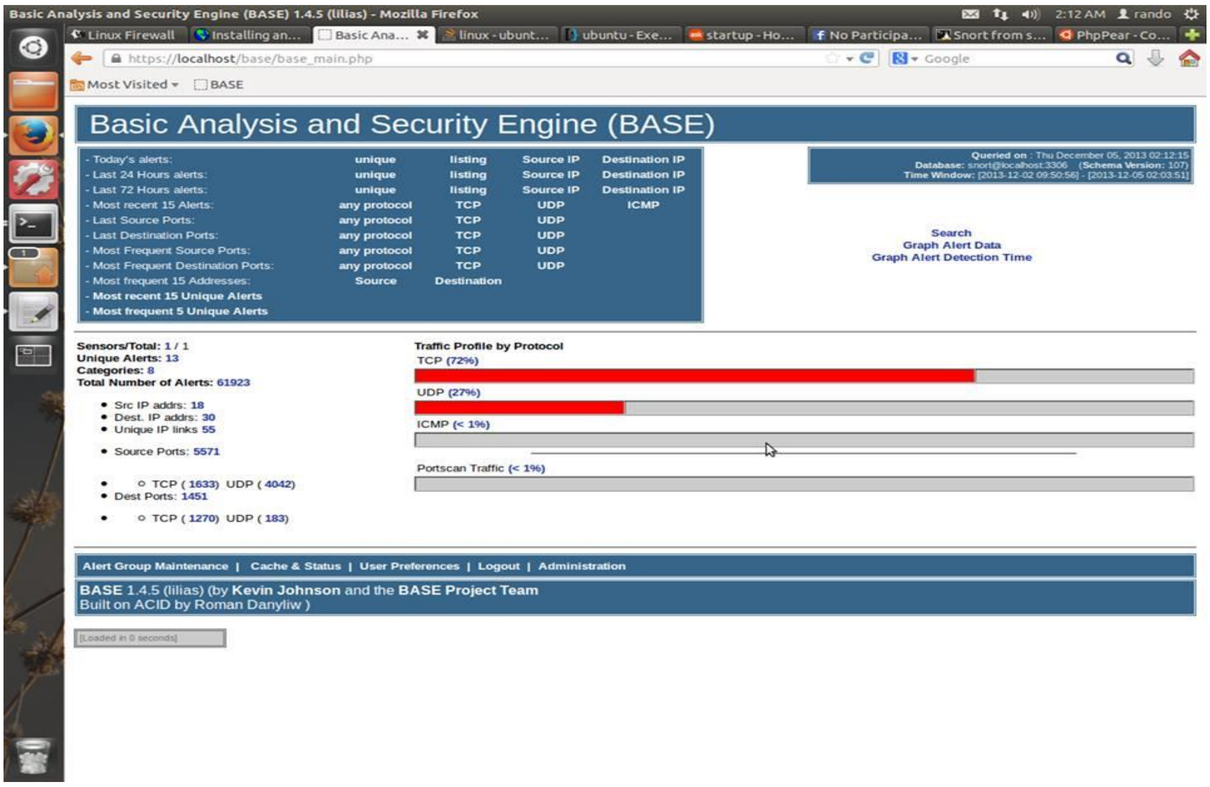
Basic Analysis and Security Engine (BASE)

Login: rando
Password: ••••••
Login Reset

BASE 1.4.5 (lilias) (by Kevin Johnson and the BASE Project Team
Built on ACID by Roman Danyliw)

Figure 10: Base login interface

After entering Username and Password we are good to go and the interface will be something like this :



Basic Analysis and Security Engine (BASE)

- Today's alerts:	unique	listing	Source IP	Destination IP
- Last 24 Hours alerts:	unique	listing	Source IP	Destination IP
- Last 72 Hours alerts:	unique	listing	Source IP	Destination IP
- Most recent 15 Alerts:	any protocol	TCP	UDP	ICMP
- Last Source Ports:	any protocol	TCP	UDP	
- Last Destination Ports:	any protocol	TCP	UDP	
- Most Frequent Source Ports:	any protocol	TCP	UDP	
- Most Frequent Destination Ports:	any protocol	TCP	UDP	
- Most frequent 15 Addresses:	Source	Destination		
- **Most recent 15 Unique Alerts**				
- **Most frequent 5 Unique Alerts**				

Queried on : Thu December 05, 2013 02:12:15
Database: snort@localhost:3306 (**Schema Version:** 107)
Time Window: [2013-12-02 09:50:56] - [2013-12-05 02:03:51]

Search
Graph Alert Data
Graph Alert Detection Time

Sensors/Total: 1 / 1
Unique Alerts: 13
Categories: 8
Total Number of Alerts: 61923

- Src IP addrs: 18
- Dest. IP addrs: 30
- Unique IP links 55
- Source Ports: 5571
- ○ TCP (1633) UDP (4042)
- Dest Ports: 1451
- ○ TCP (1270) UDP (183)

Traffic Profile by Protocol
TCP (72%)
UDP (27%)
ICMP (< 1%)
Portscan Traffic (< 1%)

Alert Group Maintenance | Cache & Status | User Preferences | Logout | Administration

BASE 1.4.5 (lilias) (by **Kevin Johnson** and the **BASE Project Team**
Built on ACID by Roman Danyliw)

[Loaded in 0 seconds]

Figure 11: BASE alerts interface

2.4.6 Snort Functions

Now that everything is configured we are good to go :

First thing to do is that Snort is started as a daemon,that means that it must be always active, working as a background service.

To achieve this snort modality we have to write on the terminal :

`snort–c /etc/snort/snort.conf –D –i eth0` for the external interface

or

`snort–c /etc/snort/snort.conf  –D  –i eth1` for the internal interface

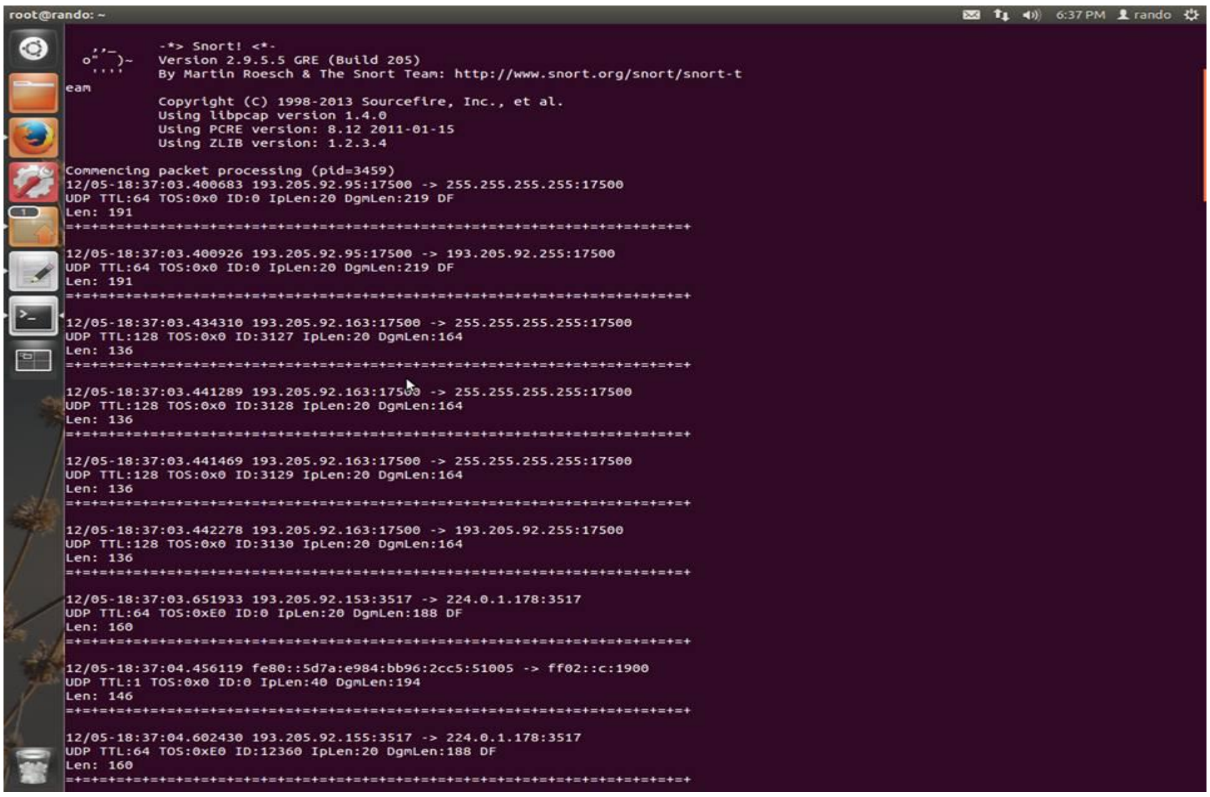
2.4.6.1 Sniffer Mode

As has been activated on both network cards in Sniffer Mode will only intercept packets, open them (not just the header) and check that their content does not correspond to an intrusion.

To visualize the traffic of TCP, UDP and ICMP packets in the terminal
It is enough to type :

`Snort –v`

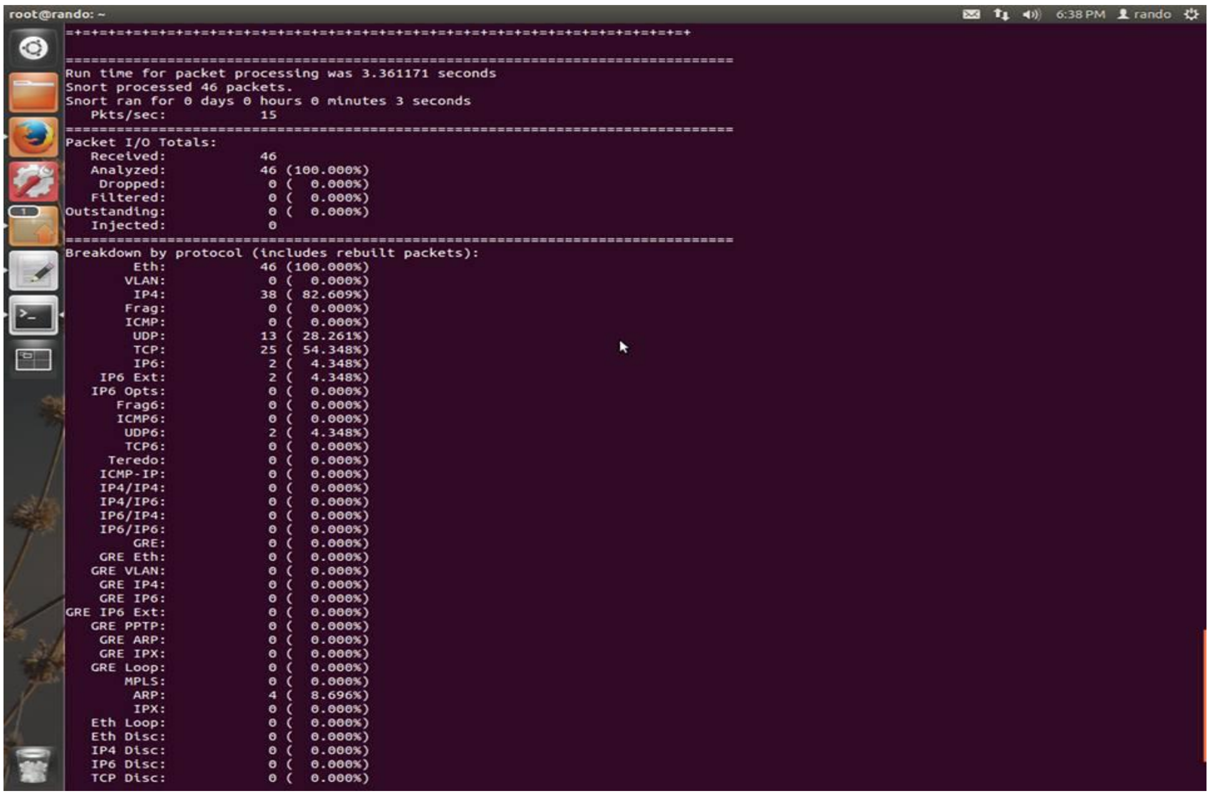
The this interface will be visualized :



```
root@rando: ~
   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.5.5 GRE (Build 205)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t
eam
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.4.0
           Using PCRE version: 8.12 2011-01-15
           Using ZLIB version: 1.2.3.4

Commencing packet processing (pid=3459)
12/05-18:37:03.400683 193.205.92.95:17500 -> 255.255.255.255:17500
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:219 DF
Len: 191
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.400926 193.205.92.95:17500 -> 193.205.92.255:17500
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:219 DF
Len: 191
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.434310 193.205.92.163:17500 -> 255.255.255.255:17500
UDP TTL:128 TOS:0x0 ID:3127 IpLen:20 DgmLen:164
Len: 136
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.441289 193.205.92.163:17500 -> 255.255.255.255:17500
UDP TTL:128 TOS:0x0 ID:3128 IpLen:20 DgmLen:164
Len: 136
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.441469 193.205.92.163:17500 -> 255.255.255.255:17500
UDP TTL:128 TOS:0x0 ID:3129 IpLen:20 DgmLen:164
Len: 136
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.442278 193.205.92.163:17500 -> 193.205.92.255:17500
UDP TTL:128 TOS:0x0 ID:3130 IpLen:20 DgmLen:164
Len: 136
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:03.651933 193.205.92.153:3517 -> 224.0.1.178:3517
UDP TTL:64 TOS:0xE0 ID:0 IpLen:20 DgmLen:188 DF
Len: 160
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:04.456119 fe80::5d7a:e984:bb96:2cc5:51005 -> ff02::c:1900
UDP TTL:1 TOS:0x0 ID:0 IpLen:40 DgmLen:194
Len: 146
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

12/05-18:37:04.602430 193.205.92.155:3517 -> 224.0.1.178:3517
UDP TTL:64 TOS:0xE0 ID:12360 IpLen:20 DgmLen:188 DF
Len: 160
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Figure 12: Snort sniffer mode

After closing of Snort, the program will provide an account of what took place, on the packets received and analyzed:



```
root@rando: ~
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

===============================================================================
Run time for packet processing was 3.361171 seconds
Snort processed 46 packets.
Snort ran for 0 days 0 hours 0 minutes 3 seconds
   Pkts/sec:           15
===============================================================================
Packet I/O Totals:
   Received:           46
   Analyzed:           46 (100.000%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            0 (  0.000%)
   Injected:            0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:           46 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:           38 ( 82.609%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:           13 ( 28.261%)
        TCP:           25 ( 54.348%)
        IP6:            2 (  4.348%)
    IP6 Ext:            2 (  4.348%)
   IP6 Opts:            0 (  0.000%)
      Frag6:            0 (  0.000%)
      ICMP6:            0 (  0.000%)
       UDP6:            2 (  4.348%)
       TCP6:            0 (  0.000%)
     Teredo:            0 (  0.000%)
    ICMP-IP:            0 (  0.000%)
    IP4/IP4:            0 (  0.000%)
    IP4/IP6:            0 (  0.000%)
    IP6/IP4:            0 (  0.000%)
    IP6/IP6:            0 (  0.000%)
        GRE:            0 (  0.000%)
    GRE Eth:            0 (  0.000%)
   GRE VLAN:            0 (  0.000%)
    GRE IP4:            0 (  0.000%)
    GRE IP6:            0 (  0.000%)
GRE IP6 Ext:            0 (  0.000%)
   GRE PPTP:            0 (  0.000%)
    GRE ARP:            0 (  0.000%)
    GRE IPX:            0 (  0.000%)
   GRE Loop:            0 (  0.000%)
       MPLS:            0 (  0.000%)
        ARP:            4 (  8.696%)
        IPX:            0 (  0.000%)
   Eth Loop:            0 (  0.000%)
   Eth Disc:            0 (  0.000%)
   IP4 Disc:            0 (  0.000%)
   IP6 Disc:            0 (  0.000%)
   TCP Disc:            0 (  0.000%)
```

Figure 13: Snort sniffer mode results

2.4.6.2 Packet Mode

In this mode, Snort exept for intercepting the traffic, it also logs the contents of the latter on the hard drive, and all those that are received will be organized into a directory / log, based on the origin of the IP.

The command we have to write is :

```
snort –vde –l /var/log/snort
```

For "logging" packets only on a specific network simply specify the range of addresses used in it:

```
snort –vde –l /var/log/snort 192.168.1.0/24
```

Once the packages have been saved, you can consult with the-r option:

```
snort –dvr packet.log
```

We can also decide what to display:

```
snort –dvr packet.log icmp
```

In this way will be shown for example only ICMP packets.

2.4.6.3 NIDS MODE

To start snort in Network Intrusion Detection System(NIDS) the command to type in the terminal will be :

```
snort –dev – l /var/log/snort –h 192.168.1.0/24 –c /etc/snort/snort.conf
```

This is the way to start Snort in NIDS mode, and start the process in its basic form, namely with:

- Active log
- Snort rules in snort.conf
- Plain text in ASCII
- Log in directory

There are seven types of Alert Mode in NIDS for the output:

- FULL (-A full)
  writes the alert in a simple format.
- FAST (-A fast)
  It is the default alert mode.
- SOCKET (-A unsock)
  Send alerts to a UNIX socket in listening to another program.
- CONSOLE (-A console)
  Sends alerts to the console in a fast format
- CMG (-A cmg)
  Generates alerts in CMG format
- NONE (-A none)
  Disables the Alert mode

2.5 Snort Preprocessors

If the behavior of the packages would be detrimental, they are sent to the detection engine that will check the pattern matching with the rules. The preprocessors can be activated, deactivated and configured through the / etc / snort.conf.

To understand the use of snort preprocessors we will examine sfportscan since this IPS system consists in blocking the connection with every ip address that tries to do a portscan on the server.

The preprocessor sfportscan, is concerned with identifying the first phase of an attack, where the attacker attempts to acquire information about the protocols and services of a host or server. This preprocessor, allows to identify any type of Portscan.

First of all we need to abilitate the preprocessor flow with which the sfportscan preprocesso interacts :

```
preprocessor flow: stats_interval 0 hash 2
```

The parameters that can be configured for the preprocessor sfportscan are:

proto { <proto> }	scan_type { <scan_type> }
sense_level { <level> }	ignore_scanners { <ip_list> }
ignore_scanned { <ip_list> }	logfile { <file> }

This is a configuration example :

```
preprocessor sfportscan:
proto { all } \
scan_type { all } \
sense_level { high }
```

From the code above it is clear that we will examine the packets belonging to all protocols, will be monitored all types of scans, and the sensor will have a high sensitivity.

2.6 Snort Detection Engine

The detection engine is the most important part of Snort. Its responsibility is to detect if any intrusion activity exists in a packet.

The rules are read into internal data structures or chains where they are matched against all packets.
If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped.
If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped.
If traffic on your network is too high when Snort is working in NIDS mode, you may drop some packets and may not get a true real-time response.

The load on the detection engine depends upon the following factors:

• Number of rules
• Power of the machine on which Snort is running
• Speed of internal bus used in the Snort machine
• Load on the network

Note that the detection system can dissect a packet and apply rules on different parts of the packet.
These parts may be:

• The IP header of the packet.

• The Transport layer header. This header includes TCP, UDP or other transport layer headers. It may also work on the ICMP header.

• The application layer level header. Application layer headers include, but are not limited to, DNS header, FTP header, SNMP header, and SMTP header. You may have to use some indirect methods for application layer headers, like offset of data to be looked for.

• Packet payload. This means that you can create a rule that is used by the detection engine to find a string inside the data that is present inside the packet.

2.7 Logging and Alerting System

Depending upon what the detection engine finds inside a packet, the packet may be used to log the activity or generate an alert. Logs are kept in simple text files, tcp-dump-style files or some other form. All of the Logs are stored under /var/log/ snort folder by default. You can use " –l " command line options to modify the location of generating logs and alerts.

Chapter 3

3.1 Tests

We cannot say the system has achieved an fair configuration until the incoming attacks are rendered useless.The condition that does not allow this to happen lies in the fact that Snort, which should detect and prevent attacks, is located on a machine potentially exposed to direct attacks. As mentioned before it would have been better that the computer on which snort is installed, had remained completely hidden, to be invisible to attackers and therefore able to carry out its own tasks undisturbed.

The attacks that have been made, however, were not particularly powerful and therefore they were easily detected and rended useless by Snort.

3.1.1 Performed attack (Portscan)

Portscan attacks were performed from 2 machines, one with Windows 8.1 OS and the other one with Linux(Ubuntu 12.04) OS.

To perform the attacks we used Nmap.

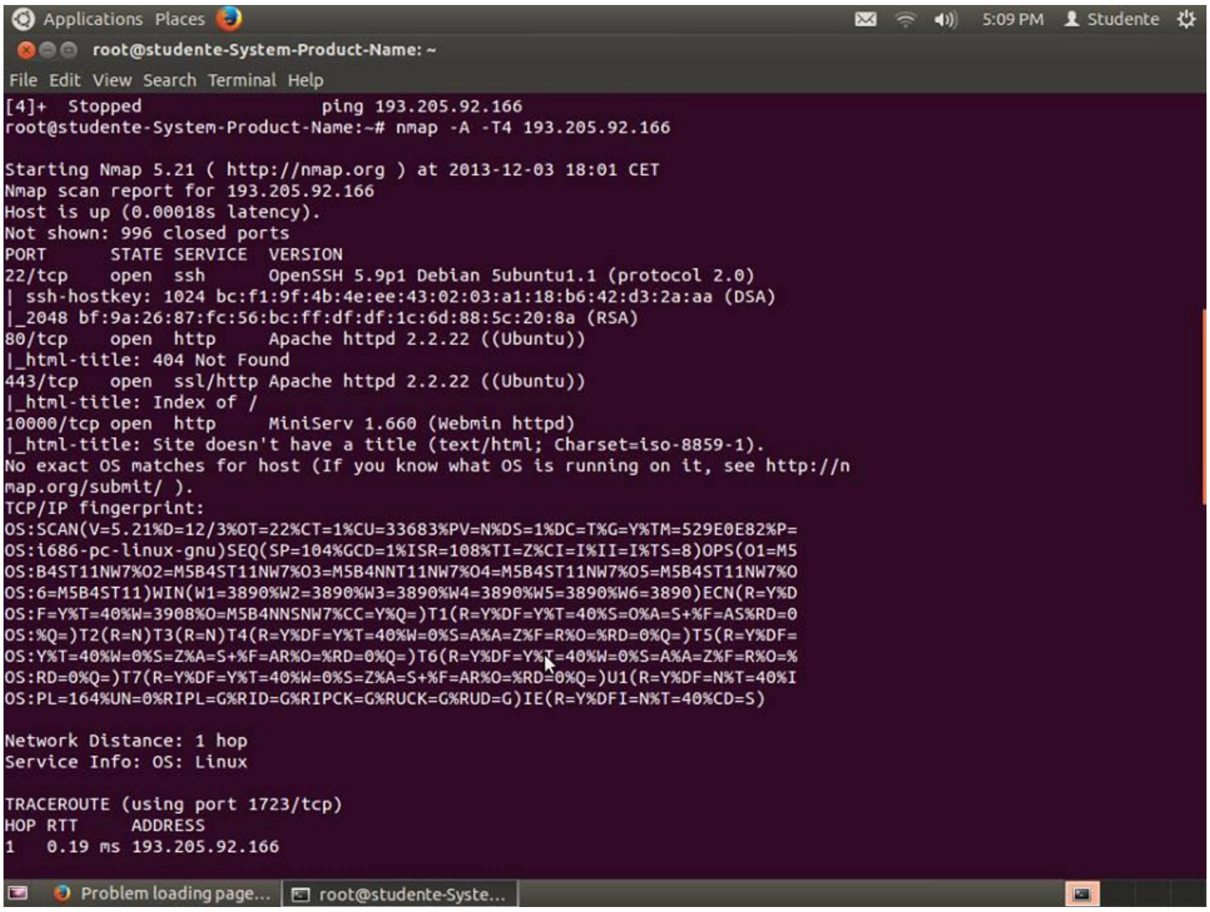
Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses.

The software provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. Nmap is also capable of adapting to network conditions
including latency and congestion during a scan. Nmap is under development and refinement by its user community.

The use of such software is very simple. The are only 2 simple steps :

- Open command prompt from Windows or the terminal fron Linux
- Type in nmap -A –T4 (ipaddress) 193.205.92.166

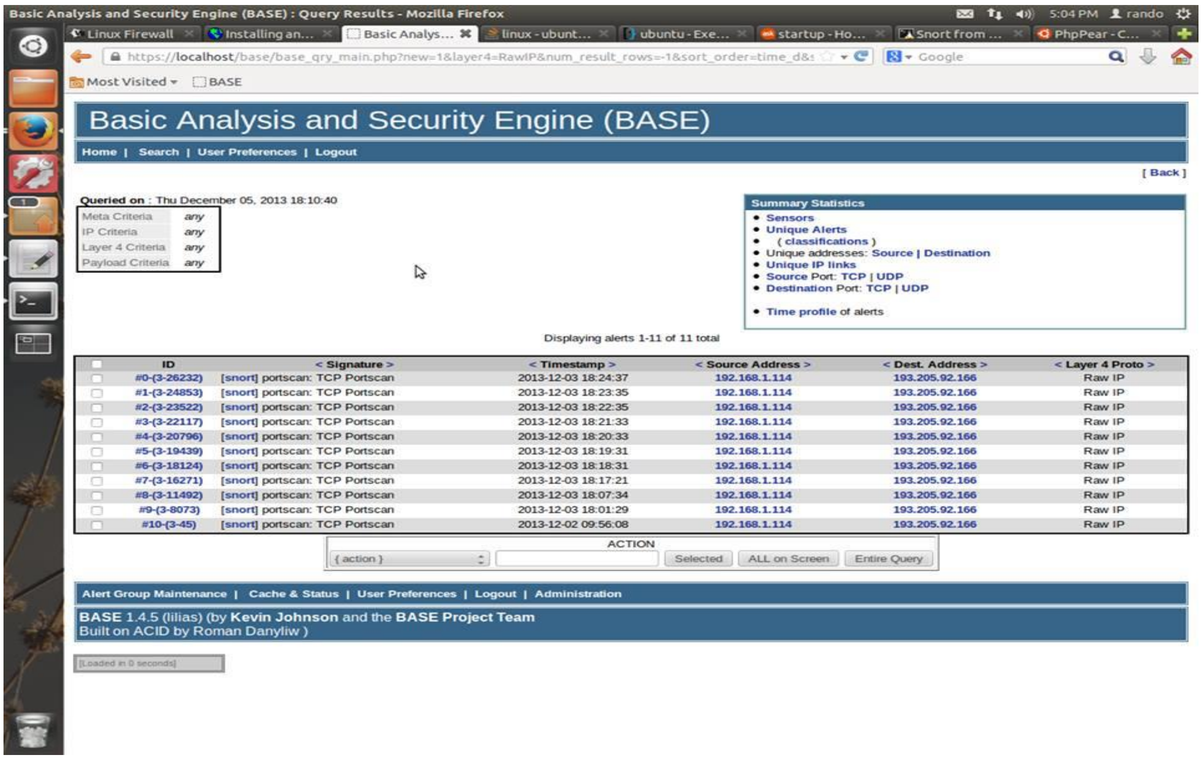
The porstcan starts and this is what the results look like(linux terminal) :



```
[4]+  Stopped                 ping 193.205.92.166
root@studente-System-Product-Name:~# nmap -A -T4 193.205.92.166

Starting Nmap 5.21 ( http://nmap.org ) at 2013-12-03 18:01 CET
Nmap scan report for 193.205.92.166
Host is up (0.00018s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.1 (protocol 2.0)
| ssh-hostkey: 1024 bc:f1:9f:4b:4e:ee:43:02:03:a1:18:b6:42:d3:2a:aa (DSA)
|_2048 bf:9a:26:87:fc:56:bc:ff:df:df:1c:6d:88:5c:20:8a (RSA)
80/tcp    open  http     Apache httpd 2.2.22 ((Ubuntu))
|_html-title: 404 Not Found
443/tcp   open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
|_html-title: Index of /
10000/tcp open  http     MiniServ 1.660 (Webmin httpd)
|_html-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
No exact OS matches for host (If you know what OS is running on it, see http://n
map.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=5.21%D=12/3%OT=22%CT=1%CU=33683%PV=N%DS=1%DC=T%G=Y%TM=529E0E82%P=
OS:i686-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=108%TI=Z%CI=I%II=I%TS=8)OPS(O1=M5
OS:B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11NW7%O
OS:6=M5B4ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)ECN(R=Y%D
OS:F=Y%T=40%W=3908%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0
OS:%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=
OS:Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%
OS:RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%I
OS:PL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux

TRACEROUTE (using port 1723/tcp)
HOP RTT     ADDRESS
1   0.19 ms 193.205.92.166
```

Figure 14: Portscan Example

With this two screenshots we can see the alerts generated by snort thanks to the interface that BASE provides us :



Basic Analysis and Security Engine (BASE) : Query Results - Mozilla Firefox
https://localhost/base/base_qry_main.php?new=1&layer4=RawIP&num_result_rows=-1&sort_order=time_d&s
Basic Analysis and Security Engine (BASE)
Home | Search | User Preferences | Logout
[Back]

Queried on : Thu December 05, 2013 18:10:40

Meta Criteria	any
IP Criteria	any
Layer 4 Criteria	any
Payload Criteria	any

Summary Statistics
- Sensors
- Unique Alerts
  (classifications)
- Unique addresses: Source | Destination
- Unique IP links
- Source Port: TCP | UDP
- Destination Port: TCP | UDP
- Time profile of alerts

Displaying alerts 1-11 of 11 total

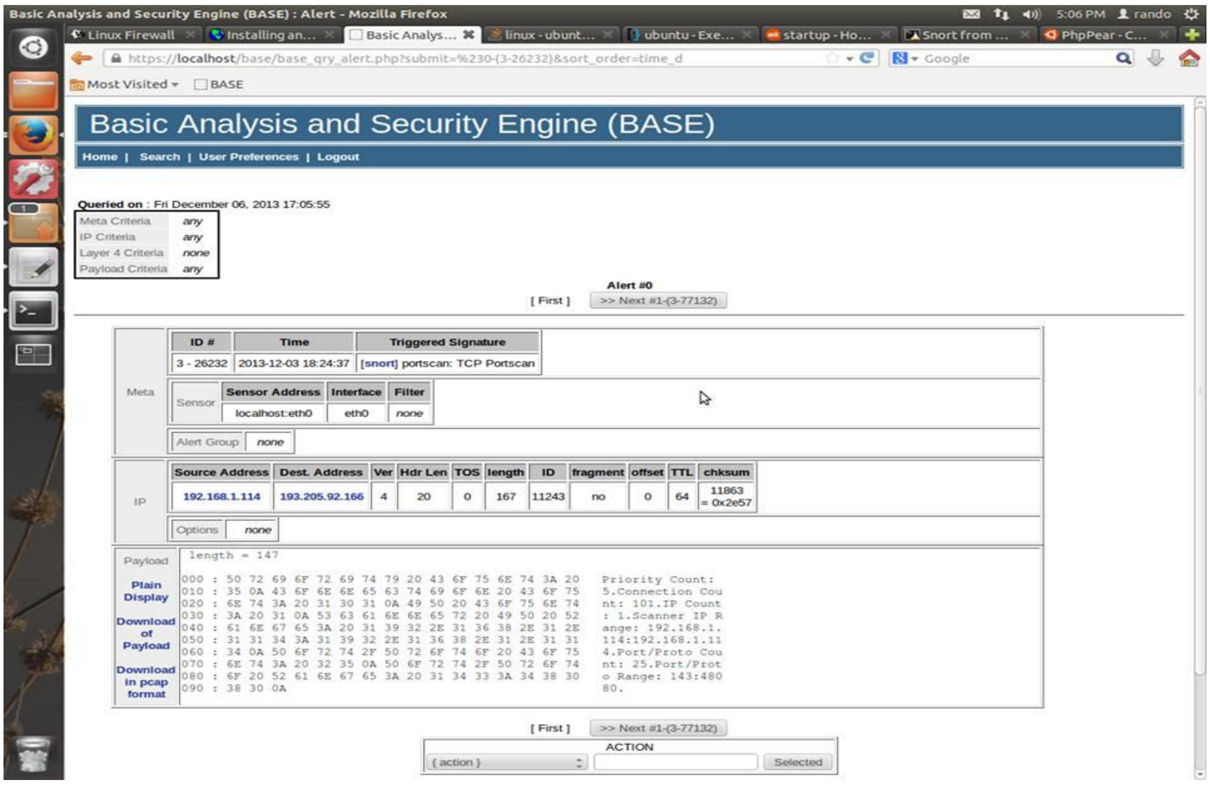
ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(3-26232)	[snort] portscan: TCP Portscan	2013-12-03 18:24:37	192.168.1.114	193.205.92.166	Raw IP
#1-(3-24853)	[snort] portscan: TCP Portscan	2013-12-03 18:23:35	192.168.1.114	193.205.92.166	Raw IP
#2-(3-23522)	[snort] portscan: TCP Portscan	2013-12-03 18:22:35	192.168.1.114	193.205.92.166	Raw IP
#3-(3-22117)	[snort] portscan: TCP Portscan	2013-12-03 18:21:33	192.168.1.114	193.205.92.166	Raw IP
#4-(3-20796)	[snort] portscan: TCP Portscan	2013-12-03 18:20:33	192.168.1.114	193.205.92.166	Raw IP
#5-(3-19439)	[snort] portscan: TCP Portscan	2013-12-03 18:19:31	192.168.1.114	193.205.92.166	Raw IP
#6-(3-18124)	[snort] portscan: TCP Portscan	2013-12-03 18:18:31	192.168.1.114	193.205.92.166	Raw IP
#7-(3-16271)	[snort] portscan: TCP Portscan	2013-12-03 18:17:21	192.168.1.114	193.205.92.166	Raw IP
#8-(3-11492)	[snort] portscan: TCP Portscan	2013-12-03 18:07:34	192.168.1.114	193.205.92.166	Raw IP
#9-(3-8073)	[snort] portscan: TCP Portscan	2013-12-03 18:01:29	192.168.1.114	193.205.92.166	Raw IP
#10-(3-45)	[snort] portscan: TCP Portscan	2013-12-02 09:56:08	192.168.1.114	193.205.92.166	Raw IP

ACTION
{ action }
Selected
ALL on Screen
Entire Query
Alert Group Maintenance | Cache & Status | User Preferences | Logout | Administration

BASE 1.4.5 (lilias) (by Kevin Johnson and the BASE Project Team
Built on ACID by Roman Danyliw)

[Loaded in 0 seconds]

Figure 15: Portscan alert in BASE



Basic Analysis and Security Engine (BASE) : Alert - Mozilla Firefox
https://localhost/base/base_qry_alert.php?submit=%230-(3-26232)&sort_order=time_d
Basic Analysis and Security Engine (BASE)
Home | Search | User Preferences | Logout

Queried on : Fri December 06, 2013 17:05:55

Meta Criteria	any
IP Criteria	any
Layer 4 Criteria	none
Payload Criteria	any

Alert #0

[First] >> Next #1-(3-77132)

Meta

ID #	Time	Triggered Signature
3 - 26232	2013-12-03 18:24:37	[snort] portscan: TCP Portscan

	Sensor Address	Interface	Filter
Sensor	localhost:eth0	eth0	none

Alert Group	none

IP

Source Address	Dest. Address	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
192.168.1.114	193.205.92.166	4	20	0	167	11243	no	0	64	11863 = 0x2e57

Options	none

Payload

Plain Display | Download of Payload | Download in pcap format

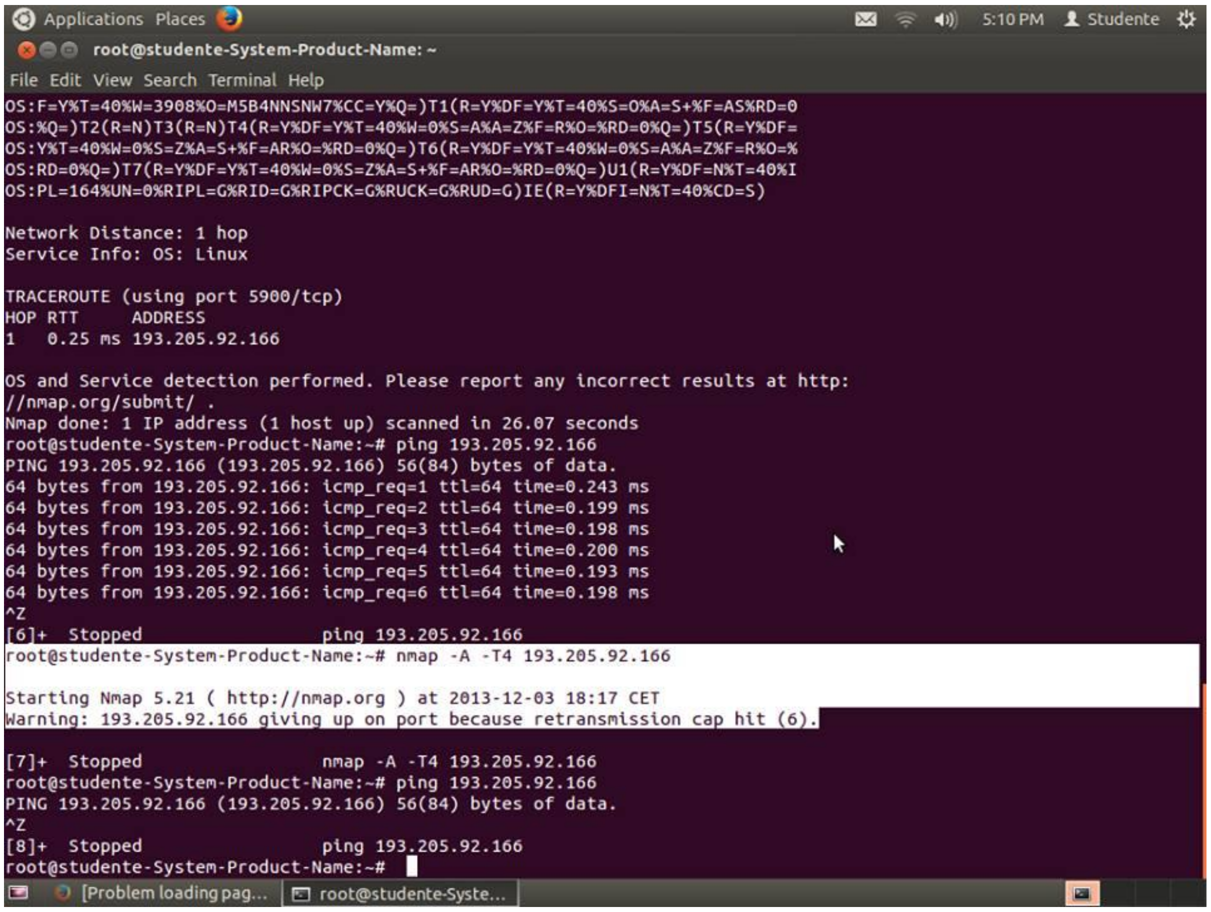
```
length = 147

000 : 50 72 69 6F 72 69 74 79 20 43 6F 75 6E 74 3A 20   Priority Count:
010 : 35 0A 43 6F 6E 6E 65 63 74 69 6F 6E 20 43 6F 75   5.Connection Cou
020 : 6E 74 3A 20 31 30 31 0A 49 50 20 43 6F 75 6E 74   nt: 101.IP Count
030 : 3A 20 31 0A 53 63 61 6E 6E 65 72 20 49 50 20 52   : 1.Scanner IP R
040 : 61 6E 67 65 3A 20 31 39 32 2E 31 36 38 2E 31 2E   ange: 192.168.1.
050 : 31 31 34 3A 31 39 32 2E 31 36 38 2E 31 2E 31 31   114:192.168.1.11
060 : 34 0A 50 6F 72 74 2F 50 72 6F 74 6F 20 43 6F 75   4.Port/Proto Cou
070 : 6E 74 3A 20 32 35 0A 50 6F 72 74 2F 50 72 6F 74   nt: 25.Port/Prot
080 : 6F 20 52 61 6E 67 65 3A 20 31 34 33 3A 34 38 30   o Range: 143:480
090 : 38 30 0A                                          80.
```

[First] >> Next #1-(3-77132)

Figure 16: Details about the portscan detected

The last screenshot show what happens after I applied the anti-portscan rules on iptables :

- iptables -A INPUT -i eth1 -p tcp -m tcp --dport 80 -m recent --name portscan --set -j LOG --log-prefix "[PORT SCAN BLOCK]"
- iptables -A INPUT -i eth1 -p tcp -m tcp --dport 80 -m recent --name portscan --set -j DROP



```
OS:F=Y%T=40%W=3908%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0
OS:%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=
OS:Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%
OS:RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%I
OS:PL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux

TRACEROUTE (using port 5900/tcp)
HOP RTT      ADDRESS
1   0.25 ms 193.205.92.166

OS and Service detection performed. Please report any incorrect results at http:
//nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.07 seconds
root@studente-System-Product-Name:~# ping 193.205.92.166
PING 193.205.92.166 (193.205.92.166) 56(84) bytes of data.
64 bytes from 193.205.92.166: icmp_req=1 ttl=64 time=0.243 ms
64 bytes from 193.205.92.166: icmp_req=2 ttl=64 time=0.199 ms
64 bytes from 193.205.92.166: icmp_req=3 ttl=64 time=0.198 ms
64 bytes from 193.205.92.166: icmp_req=4 ttl=64 time=0.200 ms
64 bytes from 193.205.92.166: icmp_req=5 ttl=64 time=0.193 ms
64 bytes from 193.205.92.166: icmp_req=6 ttl=64 time=0.198 ms
^Z
[6]+  Stopped                 ping 193.205.92.166
root@studente-System-Product-Name:~# nmap -A -T4 193.205.92.166

Starting Nmap 5.21 ( http://nmap.org ) at 2013-12-03 18:17 CET
Warning: 193.205.92.166 giving up on port because retransmission cap hit (6).

[7]+  Stopped                 nmap -A -T4 193.205.92.166
root@studente-System-Product-Name:~# ping 193.205.92.166
PING 193.205.92.166 (193.205.92.166) 56(84) bytes of data.
^Z
[8]+  Stopped                 ping 193.205.92.166
root@studente-System-Product-Name:~#
```

Figure 17: Acces Denied to the host that performed the portscan towards the server

The part in white shows that the host cannot portscan anymore the server, and since the host is connected with the server via cable it does not have anymore internet connection.

✓ Anti-PortScan Works

Chapter 4

4.1 Conclusions

From the various tests it turned out that the implementation of a system that has a reasonable level of security , using only open source software is possible.

Most of the generic attacks is recognized and blocked. Things change if it has to do with high level attacks which put a strain on the Intrusion Detection System.

From this we can deduce that computer security is almost unattainable, even though the means employed are considered among the most powerful in circulation on the Net.

The administrator has to work a lot to maintain the network as ascure as possible, and for that he has to do continuous data backup,perform updates and check the status of the system often to protect the it from external threats. It it really important the continuous monitoring of log files, looking for intrusions that may be unknown for the system.

Recently in the United States of America, network security companies are hiring extra people just to monitor their system 24/7 non stop so if any attack or problem takes place thay can be ready to withstand it.

4.2 Appendix

Below is the most important file that handles, rules, preprocessors , behaviour ecc..

4.2.1 Snort.conf :

```
#--------------------------------------------------- # VRT Rule Packages Snort.conf # # For more information visit us at: # http://www.snort.org Snort
Website # http://vrt-blog.snort.org/ Sourcefire VRT Blog # # Mailing list Contact: snort-sigs@lists.sourceforge.net # False Positive reports:
fp@sourcefire.com # Snort bugs: bugs@snort.org # # Compatible with Snort Versions: # VERSIONS : 2.9.5.5 # # Snort build options: # OPTIONS : --enable-
gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --enable-active-response --enable-normalizer --enable-reload
--enable-react --enable-flexresp3 # # Additional information: # This configuration file enables active response, to run snort in # test mode -T you
are required to supply an interface -i <interface> # or test mode will fail to fully validate the configuration and # exit with a FATAL error #-------
-------------------------------------------
################################################### # This file contains a sample snort configuration. # You should take the following steps to create
your own custom configuration: # # 1) Set the network variables. # 2) Configure the decoder # 3) Configure the base detection engine # 4) Configure
dynamic loaded libraries # 5) Configure preprocessors # 6) Configure output plugins # 7) Customize your rule set # Customize preprocessor and decoder
rule set # 9) Customize shared object rule set ###################################################
################################################### # Step #1: Set the network variables. For more information, see README.variables
###################################################

# Setup the network addresses you are protecting ipvar HOME_NET 192.168.1.0/24
# Set up the external network addresses. Leave as "any" in most situations ipvar EXTERNAL_NET !$HOME_NET
# List of DNS servers on your network ipvar DNS_SERVERS $HOME_NET
# List of SMTP servers on your network ipvar SMTP_SERVERS $HOME_NET
# List of web servers on your network ipvar HTTP_SERVERS $HOME_NET
# List of sql servers on your network ipvar SQL_SERVERS $HOME_NET
# List of telnet servers on your network ipvar TELNET_SERVERS $HOME_NET
# List of ssh servers on your network ipvar SSH_SERVERS $HOME_NET
# List of ftp servers on your network ipvar FTP_SERVERS $HOME_NET
# List of sip servers on your network ipvar SIP_SERVERS $HOME_NET
# List of ports you run web servers on portvar HTTP_PORTS
[36,80,81,82,83,84,85,86,87,88,89,90,311,383,591,593,631,801,818,901,972,1158,1220,1414,1533,1741,1830,2231,2301,2381,2809,3029,3037,3057,3128,3443,37
02,4000,4343,4848,5117,5250,6080,6173,6988,7000,7001,7144,7145,7510,7770,7777,7779,8000,8008,8014,8028,8080,8081,8082,8085,8088,8090,8118,8123,8180,81
81,8222,8243,8280,8300,8500,8509,8800,8888,8899,9000,9060,9080,9090,9091,9443,9999,10000,11371,12601,15489,29991,33300,34412,34443,34444,41080,44449,5
0000,50002,51423,53331,55252,55555,56712]

# List of ports you want to look for SHELLCODE on. portvar SHELLCODE_PORTS !80
# List of ports you might see oracle attacks on portvar ORACLE_PORTS 1024:
# List of ports you want to look for SSH connections on: portvar SSH_PORTS 22
# List of ports you run ftp servers on portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified ipvar AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24
,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path) # Note for Windows users: You are advised to make this an absolute path, # such as:
c:\snort\rules
var RULE_PATH ./rules
var SO_RULE_PATH ./so_rules
var PREPROC_RULE_PATH ./preproc_rules
# If you are using reputation preprocessor set these # Currently there is a bug with relative paths, they are relative to where snort is # not
relative to snort.conf like the above variables # This is completely inconsistent with how other vars work, BUG 89986 # Set the absolute path
appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules

################################################### # Step #2: Configure the decoder. For more information, see README.decode
###################################################
# Stop generic decode events: config disable_decode_alerts
# Stop Alerts on experimental TCP options config disable_tcpopt_experimental_alerts
# Stop Alerts on obsolete TCP options config disable_tcpopt_obsolete_alerts
# Stop Alerts on T/TCP alerts config disable_tcpopt_ttcp_alerts
# Stop Alerts on all other TCPOption type events: config disable_tcpopt_alerts
# Stop Alerts on invalid ip options config disable_ipopt_alerts
# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet # config enable_decode_oversized_alerts
# Same as above, but drop packet if in Inline mode (requires enable_decode_oversized_alerts) # config enable_decode_oversized_drops
# Configure IP / TCP checksum mode config checksum_mode: all
# Configure maximum number of flowbit references. For more information, see README.flowbits # config flowbits_size: 64
# Configure ports to ignore # config ignore_ports: tcp 21 6667:6671 1356 # config ignore_ports: udp 1:17 53
# Configure active response for non inline operation. For more information, see REAMDE.active # config response: eth0 attempts 2
# Configure DAQ related options for inline operation. For more information, see README.daq # # config daq: <type> # config daq_dir: <dir> # config
daq_mode: <mode> # config daq_var: <var> # # <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw # <mode> ::= read-file | passive | inline # <var>
::= arbitrary <name>=<value passed to DAQ # <dir> ::= path as to where to look for DAQ module so's
# Configure specific UID and GID to run snort as after dropping privs. For more information see snort -h command line options # # config set_gid:
# config set_uid:
# Configure default snaplen. Snort defaults to MTU of in use interface. For more information see README # # config snaplen: #
# Configure default bpf_file to use for filtering what traffic reaches snort. For more information see snort -h command line options (-F) # # config
bpf_file: #
# Configure default log directory for snort to log to. For more information see snort -h command line options (-l) # # config logdir:
################################################### # Step #3: Configure the base detection engine. For more information, see README.decode
###################################################
# Configure PCRE match limitations config pcre_match_limit: 3500 config pcre_match_limit_recursion: 1500
# Configure the detection engine See the Snort Manual, Configuring Snort - Includes - Config config detection: search-method ac-split search-optimize
max-pattern-len 20
# Configure the event queue. For more information, see README.event_queue config event_queue: max_queue 8 log 5 order_events content_length
################################################### ## Configure GTP if it is to be used. ## For more information, see README.GTP
####################################################
# config enable_gtp
################################################### # Per packet and rule latency enforcement # For more information see README.ppm
```

```
###################################################
# Per Packet latency configuration #config ppm: max-pkt-time 250, \ # fastpath-expensive-packets, \ # pkt-log
# Per Rule latency configuration #config ppm: max-rule-time 200, \ # threshold 3, \ # suspend-expensive-rules, \ # suspend-timeout 20, \ # rule-log
alert
################################################### # Configure Perf Profiling for debugging # For more information see README.PerfProfiling
###################################################
#config profile_rules: print all, sort avg_ticks #config profile_preprocs: print all, sort avg_ticks

################################################### # Configure protocol aware flushing # For more information see README.stream5
################################################### config paf_max: 16000

################################################### # Step #4: Configure dynamic loaded libraries. # For more information, see Snort Manual,
Configuring Snort - Dynamic Modules ###################################################
# path to dynamic preprocessor libraries dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
# path to base preprocessor engine dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
# path to dynamic rules libraries dynamicdetection directory /usr/local/lib/snort_dynamicrules
################################################### # Step #5: Configure preprocessors # For more information, see the Snort Manual, Configuring Snort
- Preprocessors ###################################################
# GTP Control Channle Preprocessor. For more information, see README.GTP # preprocessor gtp: ports { 2123 3386 2152 }
# Inline packet normalization. For more information, see README.normalize # Does nothing in IDS mode preprocessor normalize_ip4 preprocessor
normalize_tcp: ips ecn stream preprocessor normalize_icmp4 preprocessor normalize_ip6 preprocessor normalize_icmp6
# Target-based IP defragmentation. For more inforation, see README.frag3 preprocessor frag3_global: max_frags 65536 preprocessor frag3_engine: policy
windows detect_anomalies overlap_limit 10 min_fragment_length 100 timeout 180
# Target-Based stateful inspection/stream reassembly. For more inforation, see README.stream5 preprocessor stream5_global: track_tcp yes, \ track_udp
yes, \ track_icmp no, \ max_tcp 262144, \ max_udp 131072, \ max_active_responses 2, \ min_response_seconds 5 preprocessor stream5_tcp: policy windows,
detect_anomalies, require_3whs 180, \ overlap_limit 10, small_segments 3 bytes 150, timeout 180, \ ports client 21 22 23 25 42 53 70 79 109 110 111
113 119 135 136 137 139 143 \ 161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070 6665 6666 6667 6668 6669 \ 7000 8181 32770 32771 32772 32773
32774 32775 32776 32777 32778 32779, \ ports both 36 80 81 82 83 84 85 86 87 88 89 90 110 311 383 443 465 563 591 593 631 636 801 818 901 972 989 992
993 994 995 1158 1220 1414 1533 1741 1830 2231 2301 2381 2809 3029 3037 3057 3128 3443 3702 4000 4343 4848 5117 5250 6080 6173 6988 7907 7000 7001
7144 7145 7510 7802 7770 7777 7779 \ 7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914 7915 7916 \ 7917 7918 7919 7920 8000
8008 8014 8028 8080 8081 8082 8085 8088 8090 8118 8123 8180 8181 8222 8243 8280 8300 8500 8509 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 10000
11371 12601 15489 29991 33300 34412 34443 34444 41080 44449 50000 50002 51423 53331 55252 55555 56712 preprocessor stream5_udp: timeout 180
# performance statistics. For more information, see the Snort Manual, Configuring Snort - Preprocessors - Performance Monitor # preprocessor
perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000
# HTTP normalization and anomaly detection. For more information, see README.http_inspect preprocessor http_inspect: global iis_unicode_map
unicode.map 1252 compress_depth 65535 decompress_depth 65535
max_gzip_mem 104857600
preprocessor http_inspect_server: server default \ http_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK
UNLINK OPTIONS HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS
BITS_POST CCM_POST SMS_POST RPC_IN_DATA RPC_OUT_DATA RPC_ECHO_DATA } \ chunk_length 500000 \ server_flow_depth 0 \ client_flow_depth 0 \ post_depth
65495 \ oversize_dir_length 500 \ max_header_length 750 \ max_headers 100 \ max_spaces 200 \ small_chunk_length { 10 5 } \ ports { 36 80 81 82 83 84
85 86 87 88 89 90 311 383 591 593 631 801 818 901 972 1158 1220 1414 1533 1741 1830 2231 2301 2381 2809 3029 3037 3057 3128 3443 3702 4000 4343 4848
5117 5250 6080 6173 6988 7000 7001 7144 7145 7510 7770 7777 7779 8000 8008 8014 8028 8080 8081 8082 8085 8088 8090 8118 8123 8180 8181 8222 8243 8280
8300 8500 8509 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 10000 11371 12601 15489 29991 33300 34412 34443 34444 41080 44449 50000 50002 51423
53331 55252 55555 56712 } \ non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \ enable_cookie \ extended_response_inspection \ inspect_gzip \
normalize_utf \ unlimited_decompress \ normalize_javascript \ apache_whitespace no \ ascii no \ bare_byte no \ directory no \ double_decode no \
iis_backslash no \ iis_delimiter no \ iis_unicode no \ multi_slash no \ utf_8 no \ u_encode yes \ webroot no
# ONC-RPC normalization and anomaly detection. For more information, see the Snort Manual, Configuring Snort - Preprocessors - RPC Decode preprocessor
rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779 no_alert_multiple_requests no_alert_large_fragments no_alert_incomplete
# Back Orifice detection. preprocessor bo
# FTP / Telnet normalization and anomaly detection. For more information, see README.ftptelnet preprocessor ftp_telnet: global inspection_type
stateful encrypted_traffic no check_encrypted preprocessor ftp_telnet_protocol: telnet \ ayt_attack_thresh 20 \ normalize ports { 23 } \
detect_anomalies preprocessor ftp_telnet_protocol: ftp server default \ def_max_param_len 100 \ ports { 21 2100 3535 } \ telnet_cmds yes \
ignore_telnet_erase_cmds yes \ ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \ ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \ ftp_cmds {
EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \ ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \ ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT }
\ ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \ ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \ ftp_cmds { STRU SYST TEST TYPE USER XCUP
XCRC XCWD } \ ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \ ftp_cmds { XSEN XSHA1 XSHA256 } \ alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT
LPSV NOOP PASV PWD QUIT REIN STOU SYST XCUP XPWD } \ alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR STOU XMKD } \ alt_max_param_len
256 { CWD RNTO } \ alt_max_param_len 400 { PORT } \ alt_max_param_len 512 { SIZE } \ chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \ chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \ chk_str_fmt { MLSD MLST MODE NLST OPTS
PASS PBSZ PORT } \ chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \ chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \ chk_str_fmt {
XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \ chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \ cmd_validity ALLO < int [ char R int ] > \ cmd_validity EPSV < [
{ char 12 | char A char L char L } ] > \ cmd_validity MACB < string > \ cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \ cmd_validity
MODE < char ASBCZ > \ cmd_validity PORT < host_port > \ cmd_validity PROT < char CSEP > \ cmd_validity STRU < char FRPO [ string ] > \ cmd_validity
TYPE < { char AE [ char NTC ] | char I | char L [ number ] } > preprocessor ftp_telnet_protocol: ftp client default \ max_resp_len 256 \ bounce yes \
ignore_telnet_erase_cmds yes \ telnet_cmds yes
# SMTP normalization and anomaly detection. For more information, see README.SMTP preprocessor smtp: ports { 25 465 587 691 } \ inspection_type
stateful \ b64_decode_depth 0 \ qp_decode_depth 0 \ bitenc_decode_depth 0 \ uu_decode_depth 0 \ log_mailfrom \ log_rcptto \ log_filename \
log_email_hdrs \ normalize cmds \ normalize_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY } \ normalize_cmds { EXPN
HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SOML } \ normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-DRCP X-ERCP
X-EXCH50 } \ normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \ max_command_line_len 512 \
max_header_line_len 1000 \ max_response_line_len 512 \ alt_max_command_line_len 260 { MAIL } \ alt_max_command_line_len 300 { RCPT } \
alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \ alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG EMAL ESAM ESND ESOM EVFY IDENT
NOOP RSET } \ alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN ETRN DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB X-EXPS X-LINK2STATE
XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \ valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY } \
valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND SOML } \ valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT
X-DRCP X-ERCP X-EXCH50 } \ valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN XUSR } \ xlink2state { enabled }
# Portscan detection. For more information, see README.sfportscan
preprocessor sfportscan:
\ proto { all }
\ scan_type { all }
\ sense_level { high }
#proto { all } memcap { 10000000 } sense_level { low }
# ARP spoof detection. For more information, see the Snort Manual - Configuring Snort - Preprocessors - ARP Spoof Preprocessor # preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00# SSH anomaly detection. For more information, see README.ssh preprocessor ssh:
server_ports { 22 } \ autodetect \ max_client_bytes 19600 \ max_encrypted_packets 20 \ max_server_version_len 100 \ enable_respoverflow
enable_ssh1crc32 \ enable_srvoverflow enable_protomismatch# SMB / DCE-RPC normalization and anomaly detection. For more information, see
README.dcerpc2 preprocessor dcerpc2: memcap 102400, events [co ] preprocessor dcerpc2_server: default, policy WinXP, \ detect [smb [139,445], tcp 135,
udp 135, rpc-over-http-server 593], \ autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \ smb_max_chain 3, smb_invalid_shares ["C$",
"D$", "ADMIN$"]
# DNS anomaly detection. For more information, see README.dns preprocessor dns: ports { 53 } enable_rdata_overflow
# SSL anomaly detection and traffic bypass. For more information, see README.ssl preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 7801
7802 7900 7901 7902 7903 7904 7905 7906 7907 7908 7909 7910 7911 7912 7913 7914 7915 7916 7917 7918 7919 7920 }, trustservers, noinspect_encrypted
# SDF sensitive data preprocessor. For more information see README.sensitive_data preprocessor sensitive_data: alert_threshold 25
# SIP Session Initiation Protocol preprocessor. For more information see README.sip preprocessor sip: max_sessions 40000, \ ports { 5060 5061 5600 },
\ methods { invite \ cancel \ ack \ bye \ register \ options \ refer \ subscribe \ update \ join \ info \ message \ notify \ benotify \ do \ qauth \
sprack \ publish \ service \ unsubscribe \ prack }, \ max_uri_len 512, \ max_call_id_len 80, \ max_requestName_len 20, \ max_from_len 256, \
max_to_len 256, \ max_via_len 1024, \ max_contact_len 512, \ max_content_len 2048
# IMAP preprocessor. For more information see README.imap preprocessor imap: \ ports { 143 } \ b64_decode_depth 0 \ qp_decode_depth 0 \
bitenc_decode_depth 0 \ uu_decode_depth 0                                                                                                    # POP
preprocessor. For more information see README.pop preprocessor pop: \ ports { 110 } \ b64_decode_depth 0 \ qp_decode_depth 0 \ bitenc_decode_depth 0 \
uu_decode_depth 0
# Modbus preprocessor. For more information see README.modbus preprocessor modbus: ports { 502 }
# DNP3 preprocessor. For more information see README.dnp3 preprocessor dnp3: ports { 20000 } \ memcap 262144 \ check_crc
# Reputation preprocessor. For more information see README.reputation preprocessor reputation: \ memcap 500, \ priority whitelist, \ nested_ip inner,
\ whitelist $WHITE_LIST_PATH/white_list.rules, \ blacklist $BLACK_LIST_PATH/black_list.rules

################################################### # Step #6: Configure output plugins # For more information, see Snort Manual, Configuring Snort -
Output Modules ###################################################

# unified2 # Recommended for most installs # output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types

output unified2: filename snort.log, limit 128

# Additional configuration for specific types of installs # output alert_unified2: filename snort.alert, limit 128, nostamp # output log_unified2:
filename snort.log, limit 128, nostamp

# syslog # output alert_syslog: LOG_AUTH LOG_ALERT                                                                                # pcap

# output log_tcpdump: tcpdump.log

# metadata reference data. do not modify these lines include classification.config include reference.config

################################################### # Step #7: Customize your rule set # For more information, see Snort Manual, Writing Snort Rules #

# NOTE: All categories are enabled in this conf file ###################################################

# site specific rules include $RULE_PATH/local.rules

################################################### # Step #8: Customize your preprocessor and decoder alerts # For more information, see
README.decoder_preproc_rules ###################################################

# decoder and preprocessor event rules include $PREPROC_RULE_PATH/preprocessor.rules include $PREPROC_RULE_PATH/decoder.rules include
$PREPROC_RULE_PATH/sensitive-data.rules

################################################### # Step #9: Customize your Shared Object Snort Rules # For more information, see http://vrt-
blog.snort.org/2009/01/using-vrt-certified-shared-object-rules.html ###################################################

# dynamic library rules # include $SO_RULE_PATH/bad-traffic.rules # include $SO_RULE_PATH/chat.rules # include $SO_RULE_PATH/dos.rules # include
$SO_RULE_PATH/exploit.rules # include $SO_RULE_PATH/icmp.rules # include $SO_RULE_PATH/imap.rules # include $SO_RULE_PATH/misc.rules # include
$SO_RULE_PATH/multimedia.rules # include $SO_RULE_PATH/netbios.rules # include $SO_RULE_PATH/nntp.rules # include $SO_RULE_PATH/p2p.rules # include
$SO_RULE_PATH/smtp.rules # include $SO_RULE_PATH/snmp.rules # include $SO_RULE_PATH/specific-threats.rules # include $SO_RULE_PATH/web-activex.rules #
include $SO_RULE_PATH/web-client.rules # include $SO_RULE_PATH/web-iis.rules # include $SO_RULE_PATH/web-misc.rules

# Event thresholding or suppression commands. See threshold.conf include threshold.conf
```

Bibliography

Snort
http://snort.org

Understanding IPS and IDS
https://www.sans.org/reading-room/whitepapers/detection/understanding-ips-ids-ips-ids-defense-in-depth-1381

Ubuntu Server Guide
https://help.ubuntu.com/12.04/serverguide/serverguide.pdf

DHCP client-server interaction
https://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=%2Frzakg%2Frzakgconceptinteract.htm

Ubuntu 12.04 IPv4 NAT Gateway and DHCP Server
http://codeghar.wordpress.com/2012/05/02/ubuntu-12-04-ipv4-nat-gateway-and-dhcp-server/

Configuring Apache 2 on Debian, Ubuntu
http://www.control-escape.com/web/configuring-apache2-debian.html

How To Set Up a Firewall Using IP Tables on Ubuntu 12.04
https://www.digitalocean.com/community/articles/how-to-set-up-a-firewall-using-ip-tables-on-ubuntu-12-04

Webmin
http://www.ubuntugeek.com/how-to-install-webmin-on-ubuntu-13-10-raring-ringtail-server.html

Building a Debian\Snort based IDS
https://www.snort.org/assets/167/IDS_deb_snort_howto.pdf

Snort from scratch (Part II): Installing BASE & barnyard2
https://opentodo.net/2012/10/snort-from-scratch-part-ii/

Oinkmaster
http://oinkmaster.sourceforge.net/install.shtml

How to Mitigate Port Scan Attack with Iptables
http://kb.simplewallsoftware.com/how-to/how-to-mitigate-port-scan-attack-with-iptables/

Snort 2.0 intrusion detection   Brian Caswell, Jay Beale

INTRUSION DETECTION WITH SNORT Advanced IDS Techniques Using Snort, Apache, MySql, PHP, and ACID   RAFEEQ UR REHMAN