Firewall Configuration and Testing

Mohamed Amer Qamar Nazir
Email: {moham106, qamna569}@student.liu.se
Supervisor: David Byers, {davby@ida.liu.se}
Project Report for Information Security Course
Linköpings universitetet, Sweden

Abstract

Firewall is one of the important security software that protects your system from other network. Well configured firewall gives confidence to the administrator that his system is protected from malicious attacks. Realizing good firewall configuration is a matter of testing. In this project paper we will configure and test a firewall, and discover how costly can the testing process be. We also present some technologies, environments and a way of doing firewall configuration and testing using some of the basic rules and included these results in our report.

1. Introduction

Firewall is the set of rules which protects the networked computer system from unauthorized access. It can be implemented on software or hardware or both.

All messages entering or leaving to system is through firewall. It allows accessing only those messages which are allowed by the firewall and block the others which are not allowed to enter. Firewall can defend your one system from whole network or the whole network from the other network.

There is special security criteria are defined in the firewall. So every message enter to the system must meet that specified security criteria to enter that system.

In this paper we will explain how we will configure firewall on LINUX environment. After completing the configuration at first level we will test it on the same system, and then to the other systems. This report will contain the results of our tests and some of our experiences.

Further in section 2 represent the background of the project, section 3 represent some practical work, section 4 represent some of the related work in firewall implementation and finally section 5 conclude the whole report.

2. Background

Now a day's Internet is a dangerous place for your computer but it is not so few years before. Few years before people can happily do their business on the web without any protection. There is very less chance of virus attack, malware or hacking. Now it is impossible. Increasing number of Internet users also increase number viruses and malware. Many of these don't need permission to execute on your system.

To protect your computer from today Internet environment: your pc should have effective antivirus, antivirus updates and firewall. Linux operating system includes firewall protection to enhance security. Properly configured firewall can definitely increase the security of a network.

This report will cover using a Linux computer as a firewall between private network, a DMZ network and Internet. Iptables is the firewall implementation that is used throughout the configuration and testing process.

2.1 Theoretical Methods

Firewall testing is very important, it gives confidence to the administrator that the firewall rules they write are working properly. The right packet is accepted and right packet is dropped.

Firewall testing is difficult because there are many parameters which resulting is huge number of possible parameter combination.

There are number of possible combination of test cases can be used to test firewall rules. Typically each test case is viewed as a row in a table or in database term, a relation. So the main problems are:

1. Test case selection: which test cases should be applied? Different mathematical calculation can be used for it.
2. Test case execution: After selecting test cases, what method should be applied to execute it?

For first problem an efficient algorithm will be sufficient. For second there are libraries to factor out the many details of packet generation, transmission and reception [1].

Yong Du and Daniel Hoffman have presented tools and techniques for testing iptables, the methods and tools also apply to other firewall products.

Iptables are used for packet filtering based on header fields e.g., IP address, TCP and UDP port and TCP flag. Four main features of iptables are stateless filtering, state full filtering, network address/port translation, and logging.

The syntax of iptables is simple. Typically rule is ACCEPT or DROP. The rules work like C switch statement. If Pi matches, then ai is invoked. If no predicate matches, default action is make (ACCEPT or DROP). Usually default DROP is chosen for security reason. Iptables are implemented using Linux command line.

Testing configuration consist of two PCs- the driver and system under test (SUT)- The SUT configured so that traffic enter from eth1 with destination IP address and routed to eth2 and vice versa. ARP (Address Resolution Protocol) packets are used to map IP (Internet Protocol) address to MAC (Medium Access Control) addresses.

The first issue is how to generate, send, and receive frames on the drivers eth1 and eth2 interface. They have developed a raw socket library which makes the resulting test cases much easier to understand and modify. So at the end result is a simple open/close/read/write interface, much like the Linux raw I/O interface.

They created test template by providing different combination of parameters. There are three strategies they used for Tuple generation: Cartesian product generation, boundary value generation and pair wise generation. The testing framework for iptables has been implement in a tool called PBit (Pattern Based iptables tester). The useful feature of PBit is that you can modify the test configuration at run time [1].

2.2 Practical Methods

In this section we will describe the testing method implementation, like Programming language or a scripting language that is used to implement an automated test method

The test was performed using the nmap tool by executing manually a set of nmap scans and collecting results. The types of scans [6] performed were:

- TCP scan
- SYN scan
- FIN scan
- Null scan
- Xmas tree scan
- UDP scan
- IP protocol scan
- ACK Scan
- OS Fingerprinting
- Window Scan
- RPC Scan
- List Scan
- Version Detection

Each scan command was executed for each network address space, the DMZ address space and the LAN address space.

2.3 Technologies

In this section we will describe the technologies that are using in the project environment like UML ”User Mode Linux”, MLN tool and iptables

So we will start with the “UML”. UML is a port of Linux to the Linux system call interface, and allows users to run any number of virtual systems (UML instances) without the need for special privileges. The “UML” system also includes basic facilities for networking virtual machines. [4]

To simplify setup of networks of UML instances, a tool called “MLN” is use. MLN (Manage Large Networks) is a virtual machine administration tool designed to build and run virtual machine networks based on Xen, VMWare Server and User-Mode Linux. It is ideal for creating virtual network labs for education, testing, hosting or simply playing around with virtual machines. The goal is to ease the configuration and management of virtual networks. Xen and User-Mode Linux are widely used as tools for testing, learning and virtual hosting. MLN builds and configures file system templates based on its descriptive and easy programming language and stores them in an organized manner. It also generates start and stop scripts for each virtual host, enabling you to manage a running virtual network by stopping individual virtual machines within a network and starting them again. MLN makes it possible to have several separate networks, projects, at once and even connect them together to create larger networks. [5]

Ipchains are the most common firewall/Nat packages running on linux. Iptables are the enhanced product of ipchains by net filter organization.

Iptables is packet filtering firewall software. Packets inspected by iptables are passing through sequence of rules.

There are total three types of tables.

- Mangle table
- Filter queue
- Nat queue

Mangle table is responsible for alteration of quality of service bit in TCP header. This is very rarely used in satellite offices and home offices (SOHO) environment. Filter queue is responsible for packet filtering. It contains three built-in chains where we can put rules. These are:

Forward chain, Input chain and output chain. Nat queue is responsible for Network address translation. It contains two built-in chains. Which are pre-routing and post-routing? [2]

Queue Type	Queue Function	Packet Transformation Chain in Queue	Chain Function
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs. (Rarely used in SOHO environments)
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.
		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		OUTPUT	Networks address translation for packets generated by the firewall. (Rarely used in SOHO environments)

Table 1. Processing For Packet Routed By the Firewall

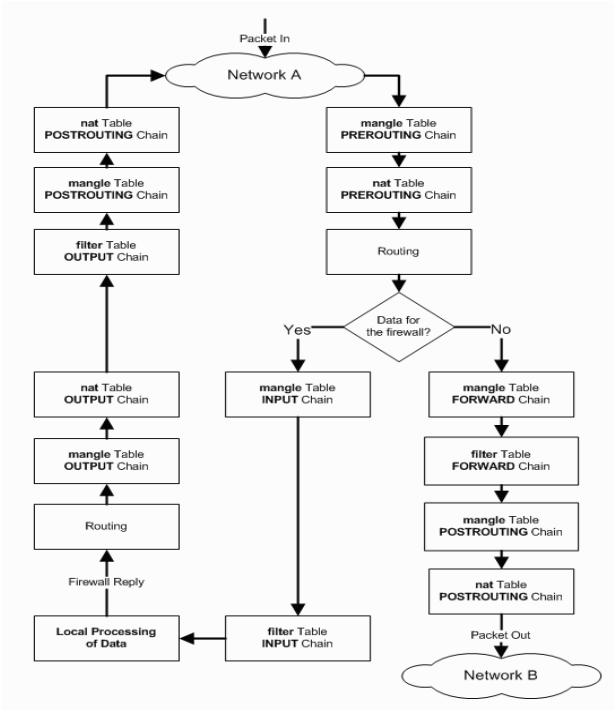

Figure 1. Iptables Packet Flow Diagram

For every firewall rule, user needs to specify iptables and ipchain. As most of the rules are related to filtering, so if user mentions any rule without an associated table then it will be considered as a part of the filter table. So we can say filter table is a default table.

In the figure 2 [2] packets arrives from Network and handled by the firewall to create a data connection.

The packet is first examined by the mangle table PREROUTING chain, then it is pass through nat table PREROUTING chain to check, it DNAT of not. Then its sent for routing. If it is defined to protected network then it passes through. FORWARD chain of mangle table for quality purpose then it is filtered by the rules of the filter table in its forward chain and if necessary it goes for SNAT in POSTROUTING chain of mangle table and then it arrives to network B. If the destination wants to apply, it will follow the same sequence.

If packet is passed through the firewall then it through INPUT chain of the mangle table, then it is filtered by a INPUT chain of the filter table. Then it passes to the firewall application for some processing.

If firewall reply then packet is sent for the routing and it is inspected by OUTPUT chain of the mangle table, if any.

Then OUTPUT chain of NAT table see if any DNAT is required then OUTPUT chain rule of filter table are applied to that packet.

Finally POSTROUTING chain of mangle checks the QoS of packet and then POSTROUTING chain of nat table check SNAT of the packet.

2.4 Environments

The environment on which we have performed the firewall configuration and testing was a linux workstation that has virtual network of virtual computers that are managed by the MLN tool. The network schema is described in the lab document. The tests are done using external and internal UML instances.

3. Solution and Analysis

3.1 Firewall testing results

After performing a testing using nmap tool and using it to scan and probe the other group firewall we found the following:

1- The firewall Mac Address which is FE:FD:00:00:ED:D6.

2- the open port, the protocol and the service of the dmz.web server:

PORT STATE SERVICE
80/tcp open http

3- The number of the up hosts on the dmz network, which are two hosts.

4- The available open services on the firewall external interface eth0, Lan interface eth1 and DMZ interface eth2 which are:

PROTOCOL STATE SERVICE
1 open icmp

5- Identified unfiltered ports on the dmz.web server which are:

PORT STATE SERVICE
80/tcp unfiltered http

6- OS detection is performed on the DMZ network with the following results:

Device type: WAP|general purpose|printer|router

Running (JUST GUESSING) : T-Home embedded (96%), Linux 2.6.X|2.4.X (94%), FON Linux 2.6.X (93%), Linksys embedded (92%), D-Link embedded (91%), Xerox embedded (90%), Enterasys embedded (89%), Netgear embedded (89%)

Aggressive OS guesses: T-Home Speedport W 501V WAP (96%), Linux 2.6.12 - 2.6.20 (94%), Linux 2.6.9 - 2.6.26 (94%), Linux 2.6.18 - 2.6.22 (93%), DD-WRT v24 (Linux 2.6.22) (93%), Linux 2.6.22 - 2.6.23 (93%), Linksys WRT300N wireless broadband router (92%), Linux 2.6.19 - 2.6.24 (Gentoo) (92%), D-Link DWL-G700AP WAP (91%), Linux 2.6.20 (91%)

7- Service detection is performed on the DMZ network with the following results:

Interesting ports on 10.19.7.10:
Not shown: 999 filtered ports
PORT STATE SERVICE VERSION
80/tcp open http Thy httpd 0.9.4 (Debian; GnuTLS 1.0.16; zlib 1.2.2)
Service Info: OS: Linux

.

4. Related work

In School of Computer Science, Telecommunication and Information System [3] they implemented the new efficient techniques for firewall testing. In their techniques they avoid exhaustive and pure random testing

Using their approach, their evaluation study shows better accuracy and performance than the random testing. Their approach is also shown to be robust as it maintains better results than random sampling even when there is a small correlation between estimated segments weight and the probability of error. When policies of different styles and segmentation sizes are implemented in their evaluation it shows that their approach is far better than all test cases. It is also proved that segmentation approach has more advantage as rule interaction.

Currently there research is still in progress. Studying different segmentation behavior for several policy styles need further investigation.

5. Conclusions

In this paper, we have explained importance of firewall, how we can configure firewall of linux based operating system using iptables but the main focus on this paper is on firewall testing. It gives confidence to the administrator that firewall is working fine and system is secure. Before that we described the usage of iptables. Then we described testing methods. The technologies we used in our project environment are UML, MLN and iptables. After applying method to some of these environments we get some results.

Our findings were that firewall testing should be given more planning and resources; because there are many variables involved in the testing process and for each variable there are many values, and for both variables and values there are many combinations to test.

References

[1] Yong Du and Daniel Hoffman. "PBit-A Pattern-Based Testing Framework for iptables", *Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04),* **IEEE** (2008),

[2] Linux Home Networking, "http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables", (28, April 2009).

[3] Adel El-Atawy, Khaled Ibrahim, Hazem Hamed, and Ehab Al-Shaer. "Policy Segmentation for Intelligent Firewall Testing", *Secure Network Protocols, 2005. (NPSec),* 1st IEEE ICNP Workshop on 6 Nov. 2005 Page(s):67 – 72.

[4] The User-mode Linux Kernel Home Page, "*www.*user-mode-linux.sourceforge.net", (30 April 2009).

[5] The Manage Large Networks project, "www.mln.sourceforge.net", (2 May 2009).

[6] Andrew J. Bennieston,"NMAP – A Stealth Port Scanner", http://nmap.org/bennieston-tutorial/, (3 May 2009).