Firewall implementation and testing

Patrik Ragnarsson, Niclas Gustafsson
E-mail: ragpa737@student.liu.se, nicgu594@student.liu.se
Supervisor: David Byers, davby@ida.liu.se
Project Report for Information Security Course
Linkpings universitetet, Sweden

Abstract

Almost every machine on the internet today is protected by a firewall. Many of them are misconfigured which is seldom discovered due to poor testing. This article summarizes our experiences in setting up and testing our own firewall, as well as performing a penetration attack from an external machine to find misconfigurations. The article also describes the tools we have used and how we used some of them.

1. Introduction

Today almost every network is protected by a firewall and people tend to heavily rely on them for security. However studies have shown that firewalls are very often misconfigured. One big reason for this is that they are often not tested thoroughly enough. This results in a situation where many people believe that their systems are protected, but in fact they are not. We believe that the problem is that today there are no good ways to systematically test the configuration of your firewall.

The purpose of this article is to present our experiences in assessing the security of a firewall that is protecting the most common internet services.

We start by explaining the steps in setting up and configuring our firewall, we then proceed into the testing of our firewall. Our configuration is then exposed to penetration testing. Our findings from testing and attacking are then summarized. The report ends with our conclusions and experiences from assessing the security of firewalls.

2. Setup

To simulate a real network set up without actually having several physical servers we have been using virtual Linux machines. Our networks consisted of 10 virtual machines, each running its own service. Three machines where placed in a demilitarized zone (DMZ) network, five was placed in a local area network (LAN), one machine acted as an external host (representing the internet) and one machine connected all these subnets and acted as a firewall.

2.1. LAN

The local area network was supposed to simulate an office network running internal services and workstations. The hosted services were web server, name server and mail server. One of the workstations utilized network address translation (NAT).

2.2. DMZ

Separated from the office network was the demilitarized zone. It housed the same services as the LAN, but accessible from external sources. The reason for separating networks in this manner is to keep external attackers limited to this subnet hence away from the LAN machines.

2.3. Firewall

Connecting all of our subnets was the firewall and routing machine. Iptables served as firewall software. The firewall was configured according to requirements given to us, these requirements can be found in Appendix A.

3. Firewall configuration

Our general design philosophy when configuring our firewall was to drop all traffic as default and make exceptions for the services running. The only exception to this was the local area network, from which we allowed any outbound traffic apart from the traffic we specifically blocked. A complete list of our firewall configuration can be found in Appendix B.

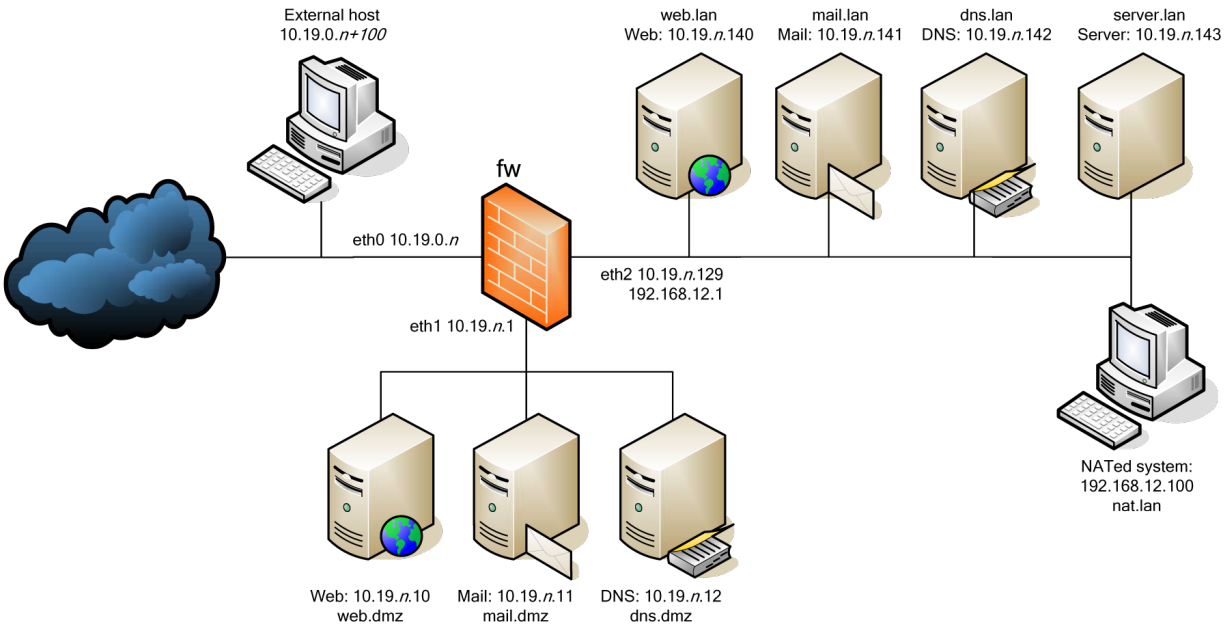

Figure 1. Overview of the network

3.1. Name server

The external name server running in the DMZ was supposed to be reachable from external sources, this was done by allowing udp packets on port 53 to pass through to the name server. The same name server was also supposed to be able to query the internal LAN name server, this was allowed in the same way. The external name server should also be able to respond to queries, this was obtained by allowing any outgoing udp packets. Since the external name server might need to query other name severs on the internet, udp packets on port 53 from the external name server to an external source on the internet was also allowed.

3.2. Mail server

To make the external mail server reachable from the internet, tcp traffic on port 25 was opened. The same port was also opened between the DMZ and the LAN, to allow the external server to connect to the internal. Hosts on the LAN was not supposed to be able to connect to any other mail server than the internal one, therefore all outgoing traffic on port 25 from the LAN was blocked, with the mail server as an exception.

3.3. Web server

The only requirement regarding web servers was that the external web server on the DMZ should be viewable from the internet. This was obtained by allowing tcp traffic on port 80 to be able to pass through from the internet to the web server in the DMZ.

3.4. ICMP

We decided to block all ICMP traffic to our subnets, however we did decide to open up a few types of ICMP packets directed at the firewall itself. The ones we decided to open up was ping and traceroute. It might be arguable if these should be open, since they might be use for several different attacks, but we felt that it did more good than harm.

3.5. Other

Other exceptions added to the firewall was tcp packets on port 22 from inside the LAN, directed at the firewall itself. This is of course to allow ssh connections to the firewall itself, this is mainly for when configuring the firewall. We also allowed the Routing Information Protocol (RIP), this was done by accepting udp packets on port 520. All traffic from the firewall, to the firewall was also opened up.

4. Tools

These are the tools we used for testing our firewall. Some of them where used solely to verify that specific rules worked, others where used in a more general way to find mistakes in our configuration.

4.1. ping

Before blocking ping-packets to the LAN and DMZ, ping was used for verifying reach-ability and that our machines where up and running. We found ping very useful during the initial phase of setting up the firewall, but it was not used very much in the later stages of testing.

4.2. telnet

Telnet was used to test the connection to our mail servers. We simulated real SMTP conversations. Telnet is very useful, but yet simple to use, to test almost any service.

4.3. tcpdump

tcpdump is used to dump all traffic on a network interface. It is possible to filter traffic in many ways, i.e. on protocol, host or port. We used it to see how far our traffic got when setting up the firewall rules. Using tcpdump we could see if we had blocked a certain service by mistake.

4.4. traceroute

We used traceroute to determine the route a packet travels across our networks. It was mainly used to test if we had correctly blocked the traceroute ICMP-traffic targeted at our subnets. We also used traceroute when we discovered a routing problem in our firewall, caused by a poorly written iptables rule regarding RIP traffic.

4.5. dig

dig is used to query DNS servers. We used it to test that our rules worked in the right way. We also used it to test the reachability of the DNS server of the other group.

4.6. nmap

nmap is a very powerful port scanner. It can scan specific hosts or entire networks. nmap includes a lot of different types of scans, for example ”TCP SYN scan” or ”SYN Stealth Scan”. nmap is also good at determine which operating system and software is running on targeted hosts. We used nmap to scanned the two subnets (LAN and DMZ) to find hosts online, and to find open ports on the hosts. It was also used to scan the firewall of the other group.

5. Testing

The goal when testing our own firewall was mainly to verify that the iptables rules worked as intended. We had a list of all the requirements and all of our rules and simply tested them one by one. When testing a service we connected to machines on all three subnets (external, DMZ and LAN). From these machines we tested if the service was usable or not using the appropriate tool described above. If the test corresponded to the firewall requirements we marked it as ”ok”, if it did not we started working on that rule again, and repeated the test. Using this method we where able to verify that all of our firewall rules worked and what was supposed to be blocked was blocked. We hoped that any mistakes we had made would be discovered by the other group while penetration testing our firewall. The requirements can be found in Appendix A.

6. Penetration Testing

We started the penetration testing against the other group by port scanning their firewall using nmap with default settings. The default scan in nmap is a ”TCP SYN scan”, it sends a SYN packet to the host on a specific port and if a SYN-ACK packet is returned, the port is open. This does not work if the target host have blocked the ICMP echo-request/response (ping) packets, which the other group had. We where then forced to use the ”SYN Stealth Scan” with ICMP pings turned off. This gave however no results, neither did any of the other scans we did of the firewall. We concluded that the firewall itself was not penetrable at this stage and moved on to the other machines.

6.1. SYN Stealth Scan

A ”SYN Stealth Scan” of their subnet (10.19.2.0/24) revealed two machines with open ports, 10.19.2.10 had port 80 open, 10.19.2.11 had port 25 open. We knew from setting up our own network that these where their public web and mail servers. We then accidentally ran a ”TCP connect() scan” on their subnet from our host machine running all the virtual servers and it revealed something interesting. We discovered another mail server 10.19.2.141 with port 25 open, this was not discovered during our initial port scan from our

own virtual machine. We tried to connect to the mail server using telnet, and it succeeded. We where not able to find a way to exploit this, but it was clearly a misconfiguration.

Now that we had the IP addresses of three of their machines, we decided to try source address spoofing attacks. These are performed by pretending to be someone else (hopefully someone with access to the targeted host) while scanning and trying to connect. We started by performing the same port scans we previously had tried, but with the built-in spoofing function of nmap. We also configured an alias on our network interface to use the spoofed IP as our own. Neither of these two ways of spoofing gave us anything new.

6.2. UDP scans

We also tried running UDP scans on the whole subnet, in hope to find open UDP ports. UDP scans works by sending empty UDP headers to ports. ICMP port unreachable error (type 3, code 3) is returned for closed ports, ICMP unreachable errors (type 3, codes 1, 2, 9, 10, or 13) is returned for filtered ports and a UDP packet is returned for open ports (or no packet at all).

This proved to be way to time consuming because of the targeted systems limiting the ICMP error message rate[1], only allowing a small number of ports being scanned per second, therefore the scanning was horribly slow. A nice feature with nmap tough, is that it detects the rate limit and slows down the scanning, so no unnecessary network traffic is sent.

7. Conclusion

When we started this project our goal was to come up with some kind of formal method for testing and assessing the security of a firewall. During the project we have realized how complicated testing a firewall really is. Firewall configurations differ so much from each other that coming up with one formal method for testing them all would be very hard.

We feel that the most important thing when testing a firewall is knowledge. Knowledge of how the protocols work, knowledge of the tools you are using and knowledge of the network you are trying to protect. With this knowledge, testing the firewall rules you have written should be pretty straight forward, but time consuming. One could make this process easier by writing a script which automates the process for future uses, but this script would still be restricted to that machine and that network.

8. Related work

8.1. Firewall Testing, 2005

A diploma thesis[2] written by Gerhard Zaugg. This is a very extensive paper on the creation of a firewall testing tool. It contains a lot of information about firewalls in general and about tools and protocols used while testing. Even though a part of the thesis is about the implementation of the testing tool, information in great detail can be found in the article. The topic of this thesis might differ a bit from ours but a lot of the tools and protocols are the same. The article describes these topics in a much lower-level and a more detailed way than we do. His conclusions are are about his own tool and therefore it is hard to compare them to ours.

8.2. A Quantitative Study of Firewall Configuration Errors, 2004

This is an IEEE article[3] written by Avishai Wool at Tel Aviv University. Avishai Wool have been leading the development of Firewall Analyzer software (www.algosec.com) for several years. In this article his focus is on Check Points FireWall-1 product (www.checkpoint.com). During two years, he have collected data from 37 firewalls. He has then analyzed the data and compiled the 12 most common misconfigurations.

In the end of the article, Avishai Wool investigate how different operating system and different firewall versions correlate to the 12 missconfigurations. A last, he draws the conclusion that smaller rule sets leads to less misconfigurations.

References

[1]: Request for Comments: 1812, Requirements for IP Version 4 Routers (section 4.3.2.8 Rate Limiting)

[2]: Gerhard Zaugg. Firewall Testing, 2005

[3]: Avishai Wool. A Quantitative Study of Firewall Configuration Errors, 2005

Appendix: A

Firewall requirements

#	Description
	General policy
1	Hosts on the Internet MUST NOT initiate connections to hosts on the LAN or DMZ, other than explicitly provided by this policy.
2	Hosts on the DMZ MUST NOT initiate connections to hosts on the LAN or the Internet, other than explicitly provided by this policy.
3	Hosts on the LAN MAY initiate connections to the Internet and DMZ, other than explicitly prohibited by this policy.
	DNS
4	Hosts on the Internet MUST be able to send DNS queries to the external DNS server.
5	Hosts on the DMZ MUST be able to send DNS queries to the internal DNS server.
6	The external DNS server MUST be able to respond to DNS queries.
7	The external DNS server MUST be able to send DNS queries to hosts on the Internet.
	Mail
8	Hosts on the Internet MUST be able to connect to SMTP on the external mail server.
9	The external mail server MUST be able to connect to SMTP on the internal mail server.
10	The internal mail server MUST be able to connect to SMTP on mail servers on the Internet.
11	Hosts on the LAN other than the internal mail server MUST NOT be able to connect to SMTP on hosts on the Internet or the DMZ.
	Web
12	Hosts on the Internet MUST be able to connect using HTTP to the external web server.
	Firewall
13	The firewall MUST be able to communicate with itself.
14	The firewall MUST accept RIP traffic from the Internet.
15	The firewall MUST accept ssh connections from the LAN.
16	The firewall MUST NOT accept any other traffic from the LAN, DMZ, or Internet.
	Other
17	The firewall must implement source NAT for LAN hosts in 192.168.12.0/24.
18	IPSec connections MUST be permitted from the Internet to the LAN.
19	Inappropriate ICMP types MUST NOT be permitted from the Internet to the LAN.
20	The firewall MUST prevent source address spoofing from the LAN as much as possible.
21	The firewall MUST prevent source address spoofing from the DMZ as much as possible.
22	The firewall MUST prevent source address spoofing from the Internet as much as possible.
23	The firewall MUST log all attempts of source address spoofing.

Appendix: B

Iptables script

Req.	Code
	Setting up variables
	IPTABLES=”/sbin/iptables” EXT_IF=”eth0” DMZ_IF=”eth1” LAN_IF=”eth2” DMZ_NET=”10.19.7.0/25” LAN_NET=”10.19.7.128/25” EXT_IP=”10.19.0.7” DMZ_IP=”10.19.7.1” LAN_IP=”10.19.7.129” WEB_DMZ_IP=”10.19.7.10” MAIL_DMZ_IP=”10.19.7.11” DNS_DMZ_IP=”10.19.7.12” WEB_LAN_IP=”10.19.7.140” MAIL_LAN_IP=”10.19.7.141” DNS_LAN_IP=”10.19.7.142” SERVER_LAN_IP=”10.19.7.143” NAT_LAN_IP=”192.168.12.100”
	$IPTABLES N LOG_SPOOF
	$IPTABLES N ICMP_PKT_IN $IPTABLES N ICMP_PKT_OUT
	$IPTABLES -A FORWARD -m state –state ESTABLISHED -j ACCEPT $IPTABLES -A FORWARD -m state –state RELATED -j ACCEPT
4	$IPTABLES -A FORWARD -i $EXT_IF -d $DNS_DMZ_IP -p UDP –dport 53 -j ACCEPT
5	$IPTABLES -A FORWARD -i $DMZ_IF -d $DNS_LAN_IP -p UDP –dport 53 -j ACCEPT
6	$IPTABLES -A FORWARD -s $DNS_DMZ_IP -p UDP -j ACCEPT
7	$IPTABLES -A FORWARD -s $DNS_DMZ_IP -p UDP –dport 53 -o $EXT_IF -j ACCEPT
8	$IPTABLES -A FORWARD -i $EXT_IF -d $MAIL_DMZ_IP -p TCP –dport 25 -j ACCEPT

9	$IPTABLES -A FORWARD -s $MAIL_DMZ_IP -d $MAIL_LAN_IP -p TCP –dport 25 -j ACCEPT
10	$IPTABLES -A FORWARD -s $MAIL_LAN_IP -o $EXT_IF -p TCP –dport 25 -j ACCEPT
11	$IPTABLES -A FORWARD -s $DNS_LAN_IP -o $DMZ_IF -p TCP –dport 25 -j DROP $IPTABLES -A FORWARD -s $WEB_LAN_IP -o $DMZ_IF -p TCP –dport 25 -j DROP $IPTABLES -A FORWARD -s $SERVER_LAN_IP -o $DMZ_IF -p TCP –dport 25 -j DROP $IPTABLES -A FORWARD -s $DNS_LAN_IP -o $EXT_IF -p TCP –dport 25 -j DROP $IPTABLES -A FORWARD -s $WEB_LAN_IP -o $EXT_IF -p TCP –dport 25 -j DROP $IPTABLES -A FORWARD -s $SERVER_LAN_IP -o $EXT_IF -p TCP –dport 25 -j DROP
12	$IPTABLES -A FORWARD -i $EXT_IF -d $WEB_DMZ_IP -p TCP –dport 80 -j ACCEPT
13	$IPTABLES -A INPUT -i lo -j ACCEPT $IPTABLES -A OUTPUT -o lo -j ACCEPT
14	$IPTABLES -A INPUT -i $EXT_IF -d $EXT_IP -p UDP –dport 520 -j ACCEPT $IPTABLES -A OUTPUT -o $EXT_IF -s $EXT_IP -p UDP –sport 520 -j ACCEPT
15	$IPTABLES -A INPUT -s $LAN_NET -d $LAN_IP -p tcp –dport 22 -m state –state NEW,ESTABLISHED –j ACCEPT $IPTABLES -A OUTPUT -p tcp –sport 22 -m state –state ESTABLISHED –j ACCEPT
17	$IPTABLES -t nat -A POSTROUTING -s 192.168.12.0/24 -o $EXT_IP -j SNAT –to-source $EXT_IP
19	$IPTABLES -A INPUT -p icmp -j ICMP_PKT_IN $IPTABLES -A OUTPUT -p icmp -j ICMP_PKT_OUT $IPTABLES -A ICMP_PKT_IN -p icmp –icmp-type echo-reply -j ACCEPT $IPTABLES -A ICMP_PKT_IN -p icmp –icmp-type echo-request -j ACCEPT $IPTABLES -A ICMP_PKT_OUT -p icmp –icmp-type echo-reply -j ACCEPT $IPTABLES -A ICMP_PKT_OUT -p icmp –icmp-type echo-request -j ACCEPT $IPTABLES -A ICMP_PKT_IN -p icmp –icmp-type destination-unreachable -j ACCEPT $IPTABLES -A ICMP_PKT_IN -p icmp –icmp-type time-exceeded -j ACCEPT $IPTABLES -A ICMP_PKT_OUT -p icmp –icmp-type destination-unreachable -j ACCEPT

	$IPTABLES -A ICMP_PKT_OUT -p icmp –icmp-type time-exceeded -j ACCEPT
20	$IPTABLES -A INPUT -i $LAN_IF -s ! $LAN_NET -j LOG_SPOOF
21	$IPTABLES -A INPUT -i $DMZ_IF -s ! $DMZ_NET -j LOG_SPOOF
22	$IPTABLES -A INPUT -i $EXT_IF -s $LAN_NET -j LOG_SPOOF $IPTABLES -A INPUT -i $EXT_IF -s $DMZ_NET -j LOG_SPOOF
23	$IPTABLES -A LOG_SPOOF -j LOG $IPTABLES -A LOG_SPOOF -j DROP
1,2	$IPTABLES -P INPUT DROP $IPTABLES -P OUTPUT DROP $IPTABLES -P FORWARD DROP
3	$IPTABLES -A FORWARD -i $LAN_IF -o $EXT_IF -j ACCEPT $IPTABLES -A FORWARD -i $LAN_IF -o $DMZ_IF -j ACCEPT