Global Information Assurance Certification Paper

SANS

GCFW - Practical Assignment

Firewalls, Perimeter Protection and VPNs
Version 1.6a

Prepared by: Roel Schouten
Date: March 2002

Table of contents

Introduction

This is the practical assignment for the SANS GCFW certification program. The first part of the document describes the network security implementation of GIAC Enterprises, an e-business, which deals in the online sale of fortune cookie sayings. This part of the assignment is divided into 3 parts:

- Assignment 1 - Security Architecture
- Assignment 2 - Security Policy
- Assignment 3 - Audit Security Architecture

The last part of the document describes the evaluation and "attack" of a SANS GCFW practical assignment previously submitted by a training participant.

1 Company

GIAC enterprises is a fairly new company and is looking for a secure network implementation to support their business. The transactions by the various user groups connected with GIAC, has to be secure, user-friendly and stable.
Along with supporting the business as described above, GIAC's network has to be protected from malicious and inexperienced users.

2 Assumptions

2.1 Budget

As GIAC is a fairly new e-business company, the security budget is rather low. This is taken into account in the design of the security architecture.

2.2 IP address space

2.2.1 External

GIAC has a class C subnet to its disposal for the external address space. The IP address range is: 197.22.1.0 - 197.22.1.255.

Author's note: This range is reserved by the IANA[1] and as far as I know it is not officially in use. I use it as an example for this assignment.

2.2.2 Internal

GIAC's internal IP address range is the class C subnet: 192.168.1.0-192.168.1.255. This is a private non-routable address space as per RFC 1918[2].

[1] See: IANA's Internet Protocol V4 Address Space: http://www.iana.org/assignments/ipv4-address-space
[2] See: RFC 1918 - Address Allocation for Private Internets: http://www.ietf.org/rfc/rfc1918.txt

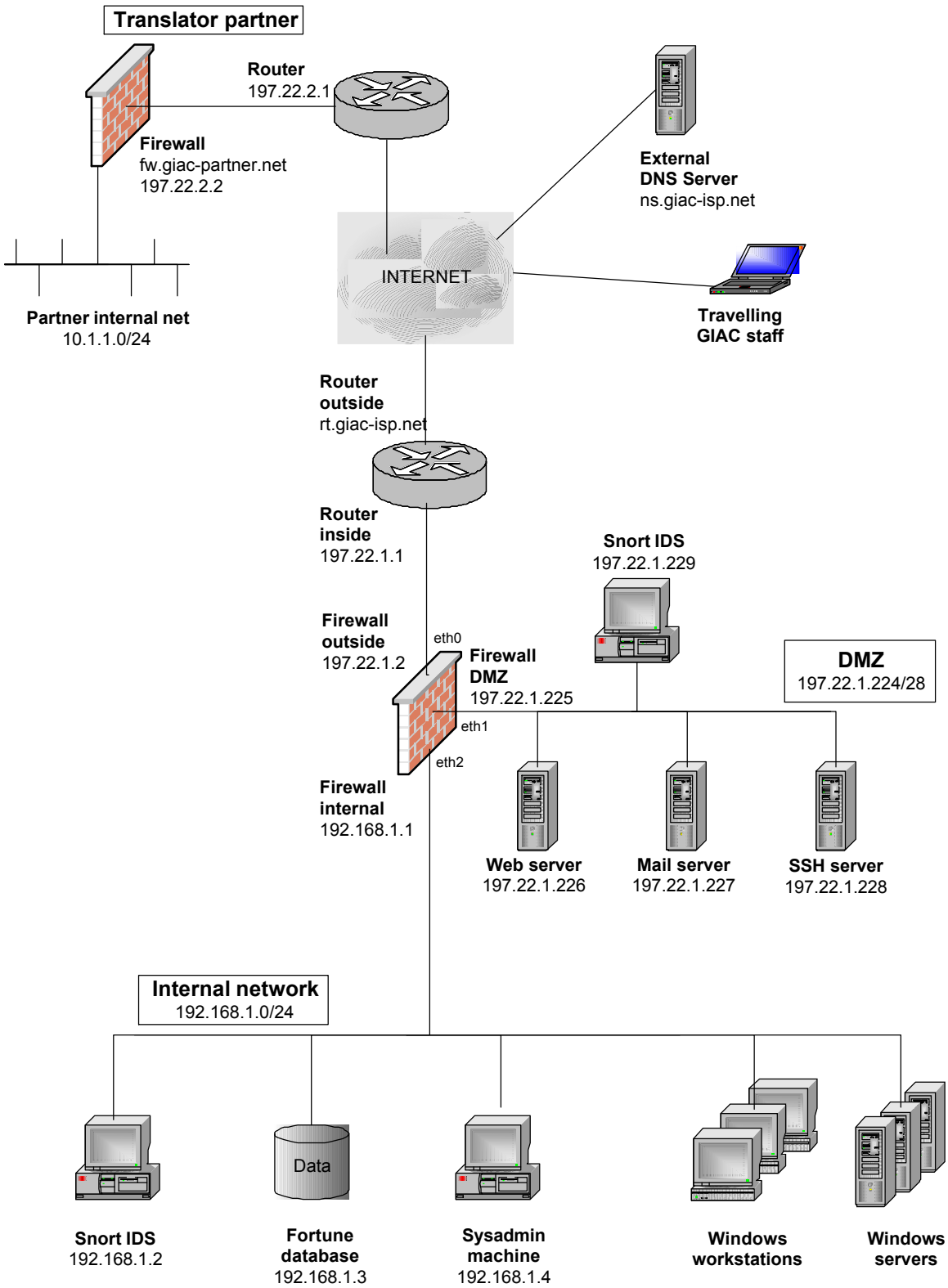
1.3 Name Service

Considering the low amount of external systems and IP addresses GIAC uses, the DNS service is managed by a third party (e.g. an ISP). This party has regular security audits of their systems.

Author's note: The name for the external DNS server that is used is ns.giac-isp.net. This name doesn't exist at the time this document was created.

Assignment 1 - Security Architecture

1 Network Drawing



2 Subnets and Systems

Below the description of the systems and subnets in GIAC's network. Most of the systems have no remote management enabled to limit the number of open services, thus reducing the security risks.

2.1 Exterior

2.1.1 Border Router

Hardware: Cisco 3620
Software: IOS 12.2
Function: - Routing Internet traffic to and from GIAC Enterprises
- Static packet filter
Services: None
Notes: Router management is done through telnet

The Cisco 3620 is chosen because it is relatively cheap (app. $4000) and can handle GIAC's current bandwidth need. It can be expanded with an extra network module if necessary in the future.
Besides routing traffic, the router acting as a static packet filter, which has a lower impact on the performance. A packet filter can easily be configured with IOS 12.2 (and other IOS's for that matter).

2.1.2 Firewall

Software: - RedHat Linux 7.2[3] - Kernel 2.4.9-21
- Netfilter/iptables 1.2.5[4]
- FreeS/WAN 1.95[5] Freeware VPN solution
Function: - Stateful firewall
- Provide VPN access to Fortune database for translators
Services: None
Notes: Firewall management is done on the firewall console

Linux and Netfilter/iptables comply with GIAC's requirements: secure, powerful, easy to manage and above all cheap (=free). Netfilter has many features and plug in possibilities, which are useful now and maybe also for future expansions.
The FreeS/WAN is a solid, free(!) VPN solution for Linux, which supports a variety of IPSec implementations.

[3] See: RedHat's web site: http://www.redhat.com/
[4] See: The Netfilter web site: http://www.netfilter.org/
[5] See: The web site of FreeS/WAN - an implementation of IPSEC & IKE for Linux: http://www.freeswan.org/

2.2 DMZ

The DMZ is subnetted in the following way: 197.22.1.224/28

- Network: 197.22.1.224
- Broadcast: 197.22.1.239
- Subnet mask: 255.255.255.240
- Host address range: 197.22.1.225-238

This makes routing set up easier for the Linux firewall. The following static route is added to the Firewall:

```
route add -net 197.22.1.224 netmask 255.255.255.240 dev
      eth1
```

Where eth1 is the interface on the DMZ.

Also an extra route to the router configuration is added, which assures that traffic to the DMZ subnet is sent to the Firewall's external interface.

2.2.1 IDS

Software: - RedHat Linux 7.2 - Kernel 2.4.9-21
- Snort 1.8.3-1[6]

Function: - Sniffing the traffic patterns on the DMZ
- Logging unusual or unexpected patterns
- Alerting in case of malicious attacks

Services: None

Notes: Snort management is done on the Linux console.

Snort is one of the strongest IDS systems available and (once again) it's free.

2.2.2 SSH server

Software: - RedHat Linux 7.2 - Kernel 2.4.9-21
- OpenSSH 3.02p1 release 1[7]

Function: Provide access to fortune sayings for customers, resellers and suppliers.

Services: TCP 22

Notes: - Management is done on the Linux console
- Only SSH2 is enabled

The SSH server is used as the Fortune distribution system. FTP could have been chosen instead, but due to the security risks posed by the FTP protocol (e.g. clear text passwords) SSH is used instead.
For easier downloading and uploading, clients can use SFTP[8]
The SSH protocol that is used is SSH2 and with Blowfish encryption.

[6] See: The Snort web site: http://www.snort.org/
[7] See: The OpenSSH web site: http://www.openssh.com/portable.html
[8] See: SSH Communications Security for a graphical SFTP client for Windows: http://www.ssh.com/products/ssh/download.cfm

Blowfish is used because it is secure, unpatented, license-free, and available free(!) for all uses[9]. GIAC would show good manners by informing Bruce Schneier, the designer of Blowfish, about their commercial use of Blowfish.

2.2.3 Mail Server

Software: - RedHat Linux 7.2 - Kernel 2.4.9-21
- Sendmail 8.12.2[10]
- Qpopper 4.0.3[11]
- OpenSSH 2.9p2 release 12

Function: - Sending and delivering e-mail to and from GIAC Enterprises
- Giving local and traveling GIAC employees possibility to read their e-mails

Services: TCP 22, 25 and 110

Notes: - Sendmail, Qpopper and SSH management is done on the Linux console
- POP3 (TCP 110) will only be listening to localhost because it is tunneled through SSH

Sendmail, OpenSSH and Qpopper are robust and easy to set up. Older versions of these software packages had multiple vulnerabilities, but these have been solved now.

2.2.4 Web Server

Software: - RedHat Linux 7.2 - Kernel 2.4.9-21
- Apache 1.3.23[12]

Function: Presenting GIAC's web site to the outside world.

Services: TCP 80

Notes: - Web server management is done on the firewall console.
- The web site is only used for company presentation and contact information. Therefore, the web site only contains static HTML pages without scripts and forms, which minimizes the risk of security breaches

2.3 Internal Network

The internal network is a combination of a Windows 2000 domain and Linux based systems.

2.3.1 IDS

Software: - RedHat Linux 7.2 - Kernel 2.4.9-21
- Snort 1.8.3-1

[9] See: Counterpane Labs - The Blowfish Encryption Algorithm: http://www.counterpane.com/blowfish.html
[10] See: The Sendmail Consortium web site: http://www.sendmail.org/
[11] See: Eudora's web site: http://www.eudora.com/qpopper/
[12] See: Apache's web site: http://www.apache.org/

Function: - Sniffing the traffic patterns on the DMZ
- Logging unusual or unexpected patterns
- Alerting in case of malicious attacks

Services: None

Notes: Snort management is done on the Linux console.

2.1.2 Fortune Database

Software: - Windows 2000 Server SP2 + latest patches
- SQL server (e.g. Microsoft's SQL server)

Function: Store fortunes

Notes: Only GIAC staff and translators involved with updating fortunes have access to this system

2.1.3 Workstations

Software: - Windows 2000 Workstation SP2 + latest patches
- ZoneAlarm Pro 2.6 personal firewall[13]
- McAfee VirusScan Professional 6.0[14]

Function: Desktops for local GIAC employees and laptops for traveling staff.

Notes: - The personal firewall and virus scanner can not be disabled by the users.
- The personal firewall and virus scanner are automatically updated each time the workstation is connected to the domain.

2.1.4 System Administrator Machine

Software: - Windows 2000 Workstation SP2 + latest patches
- ZoneAlarm Pro 2.6 personal firewall
- McAfee VirusScan Professional 6.0

Function: - Administration of the Windows 2000 domain.
- Downloading of software updates and patches from the Internet

Notes: - The personal firewall and virus scanner can not be disabled by the users.
- The personal firewall and virus scanner are automatically updated each time the workstation is connected to the domain.

2.1.5 Other Servers

Software: - Windows 2000 Server SP2 + latest patches

Function: Provide internal NetBIOS network.

3 User Groups

This section gives a general description of the various user groups

[13] See: Zonelab's web site for ZoneAlarm: http://www.zonealarm.com/
[14] See: McAfee's web site: http://www.mcafee.com/

connected with GIAC Enterprises. It explains each group's business operations with GIAC and the access requirements and restrictions to GIAC's network connected with these operations. This access is systematically described and technically implemented in Assignment 2 - Security Policy.

3.1 Customers

Definition: Companies that purchase bulk fortunes over the Internet.

GIAC only sells to users or companies that order at least 500 fortunes per month for a period of at least 1 year. Smaller customers are managed by resellers (see 3.3 - Partners).

GIAC's customers can download the fortunes from a SSH server that is located in the DMZ (see 2.2 - DMZ), which they connect to through SSH2. Customers get a unique and unchangeable strong login name and password that only gives read access to their personal directory on the SSH server (e.g. */home/customer1*). They also receive the server's public key for authentication. Both login credentials and public key are sent in separate PGP encrypted and signed e-mails. If the customer doesn't have PGP, they receive the items on a diskette sent through courier.
A random password generator generates the login names and passwords.

The customer directories on the SSH server are updated with the newest fortunes each month from the Fortune database on the internal network. In this way it is easy to make sure that each customer only receives the number of fortunes they paid for.

The customers can copy files from the SSH server using an SSH secure copy program or an SFTP client[15].
Obviously the customers must have the SSH server's public key in place before connecting.

The fortunes are placed in plain text files that are gathered in zip-files to save disk space on the SSH server.

3.2 Suppliers

Definition: The authors of fortune cookie sayings that connect to supply fortunes.

The authors get access to the SSH server on the DMZ in the same way as the customers do. Each of them gets a unique login and directory (we don't want them to "steal" other's fortunes - they can buy them if they want to have them). However, the suppliers will have write access to their directories, otherwise they will not be able to upload their fortunes.

[15] See: SSH Communications Security's web site: http://www.ssh.com/products/ssh/download.cfm

Authors submit the fortunes in plain text files that are gathered in zip-files. After submission the author sends and e-mail to GIAC and the internal Fortune database is updated with the submitted fortunes.

3.3 Partners

3.3.1 Resellers

Definition: The international partners that resell fortunes.

Resellers are considered customers and security-wise they are treated as such. See 3.1 - Customers.

3.3.2 Translators

Definition: The partners that edit, correct and translate fortunes.

Editors need to have access to the Fortune database on the Internal network to be able to do their work. Therefore a VPN connection to the Fortune database is set up for these people.

The VPN connection is realized with FreeS/WAN, an open-source VPN solution for Linux. The VPN is an IPSec implementation with IKE and ESP. See Assignment 2 - Security Policy, section 3 - VPN for a detailed explanation of the VPN implementation for one partner.

3.4 GIAC Enterprises Employees

Definition: The employees located on GIAC's internal network or traveling staff.

The internal GIAC employees can browse the Internet using only HTTP and HTTPS. To do so, they will also need DNS of course.
They read their mail from the Mail server on the DMZ using POP3 tunneled through SSH2.
Traveling staff can read and send e-mail using the Mail server on the DMZ. They also use POP3 tunneled through SSH2.

To set up the SSH tunnel, employees (traveling and local) use a Windows SSH client[16] and run the following command from a command prompt (the Mail server's public key is put on the employee machines when the IT departments installs them):

```
ssh -C -l <username> -L 110:197.22.1.227:110
     197.22.1.227
Password:
```

This script might be automatically started from the domain logon script or the start up folder in Windows.

[16] GIAC uses the client from SSH Communications Security: http://www.ssh.com/products/ssh/download.cfm

Note: The employees should be made aware of warning messages regarding authentication of the Mail server. If the public key of the Mail server (which is on the employee machines) does not match with the server's key, the employees should CLOSE the connection!

The result of the script is that it opens TCP port 110 on the local host, which is then tunneled over SSH to TCP port 110 (POP3) on the mail server on the DMZ (197.22.1.227).
Now, the e-mail client on the employee's machine has to be set up to connect to localhost (or 127.0.0.1) for receiving mails.

The reason why SSH tunneling is used is the fact that the POP3 server is located on the DMZ. If a malicious user succeeds in installing a sniffer on the DMZ, he/she will not be able to pick up password in any way.

Sending mails over SMTP is not tunneled since no passwords are exchanged here and the mail will be delivered unencrypted anyway when it is send from GIAC's mail server to another and vice versa.

Some employees on the Internal network have access to the SSH server on the DMZ for updating fortunes. They get a similar type of access as customers do; the only difference is that employees have access to all customer directories.
These employees will also have access to the Fortune database.

1.5 Regular Internet User

Definition: The people that browse the GIAC web site.

These users only need HTTP access to the Web server on the DMZ.

Assignment 2 - Security Policy

1 Border Router

The Border router is set up as a static packet filter, which has a lower impact on the router performance than stateful filtering. Stateful filtering is done by the Firewall.
Source address verification (e.g. avoiding IP address spoofing) is done with Cisco IOS's Standard IP access list because it is faster than the Extended IP access verification[17].
All other packet filtering is done with the Extended access list.
Reflexive access is not used because of its impact on the router performance.

1.1 General set up

- Disable SNMP service.
  SNMP is not used because of the security risks it induces.
- Disable other unnecessary services.
  Finger, echo, discard, chargen, and daytime services.
- Disable servers.
  Bootp and HTTP.
- Deny IP source-routing.
  IP source-routing is not needed and it can give a malicious user useful information on GIAC's network.
- Deny direct broadcasts.
  Denial of Service attacks like Smurf[18] and network mapping.
- Disable ICMP unreachables.
  ICMP unreachable message give away information on GIAC's network.
- Encrypt management password.
  The password should not be stored in clear text.
- Allow remote management.
  The System Administrator machine has access to the telnet interface on the router from the inside network. The machine's address is NAT'ed by the firewall, so therefore the external address of the Firewall is allowed to access the telnet interface.

1.2 Inbound (Ingress) Packet Filtering

- Deny GIAC IP source addresses.
  All inbound packets with source address 197.22.1.x are denied. This prevents spoofed attacks.
- Deny loop back source address.
  All inbound packets with source address 127.0.0.x are denied. This prevents spoofed attacks on the loop back address.

[17] See: GIAC GCFW section 2.3.1: Cisco Routers (fw_31_cisco.pdf).
[18] See: CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks: http://www.cert.org/advisories/CA-1998-01.html

- Deny private address spaces.
  All inbound packets with source addresses 10.x.x.x, 172.16.x.x - 172.31.x.x and 196.168.x.x are denied. Private addresses should never be seen out on the Internet, so when the router receives these, it cannot be legal traffic.
- Permit anything else.

1.3 Outbound (Egress) Packet Filtering

- Allow GIAC IP source addresses.
  All outbound packets with source address 197.22.1.x are allowed.
- Deny anything else.
  Any other source IP address than GIAC's should not be there in the first place and it should in any case not be able to connect out to the Internet.

1.4 Implementation

The Cisco IOS commands[19] that are written at the router's management interface for implementing the rules described in the previous sections. Commands are printed in `courier` font.

1.4.1 General Set Up

- Disable SNMP service:
```
no snmp
```

- Disable other unnecessary services:
```
no service tcp-small-servers
no service udp-small-servers
no service finger
```

- Disable servers:
```
no ip http
no ip bootp
```

- Deny IP source-routing:
```
no ip source-route
```

- Deny direct broadcasts:
```
no ip direct-broadcast
```

- Disable ICMP unreachables:
```
no ip unreachables
```

[19] See: Cisco IOS Configuration Fundamentals Command Reference, Release 12.2 and Cisco IOS Security Command Reference, Release 12.2:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/ffun_r/index.htm
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_r/index.htm

- Encrypt management password:
```
service password-encryption
```
- Allow remote management:
```
access-list 11 permit host 197.22.1.2
     line vty 0 4
      access-class 15
      login
```

1.4.2 Inbound (Ingress) Packet Filtering

After implementing the filters below, the following command is run to apply the filters to incoming traffic from the Internet to the router:

```
interface serial 0
ip access-group 10 in
```

- Deny GIAC IP source addresses:
```
access-list 10 deny 197.22.1.0 0.0.0.255 any
```

- Deny loop back source address:
```
access-list 10 deny 127.0.0.0 0.255.255.255 any
```

- Deny private address spaces:
```
access-list 10 deny 10.0.0.0 0.255.255.255 any
access-list 10 deny 172.16.0.0 0.15.255.255 any
access-list 10 deny 192.168.0.0 0.0.255.255 any
```

- Permit anything else:
```
access-list 10 permit any
```

1.4.3 Outbound (Egress) Packet Filtering

After implementing the filters below, the following command is run to apply the filters to traffic going from GIAC's network to the Ethernet interface of the router:

```
interface ethernet 0
ip access-group 20 in
```

- Allow GIAC IP source addresses:
```
access-list 20 permit 197.22.1.0 0.0.0.255 any
```

- Deny anything else:
  This actually doesn't have to be specified, because the router does this automatically when one or more ACL's are specified.

2 Firewall

2.1 General set up

Disable all open services on the Firewall. No daemons whatsoever should be running (e.g. portmapper, lpd, sendmail etc. - services that are installed and enabled by default in RedHat). Best practice would be to remove these services completely from the Linux operating system on the Firewall. When using X to configure the Firewall, make sure to use the *-nolisten tcp* option when starting the X desktop (this option can also be entered in the *serverargs=""* statement in the *startx* script).

2.2 Internet to DMZ

- SSH access for customers, suppliers and resellers to the SSH server.
- SSH access for traveling GIAC employees to the Mail server for reading e-mail. POP3 is tunneled through SSH.
- SMTP access for traveling GIAC employees to the Mail server for sending e-mail.
- SMTP access for external mail servers to the Mail server for delivery of mail sent to GIAC.
- HTTP access to the Web server for regular Internet users.

2.3 DMZ to Internet

- SMTP access from the Mail server to external mail servers for delivery of mail going from GIAC to the Internet.
  Note: POP3 is NOT open to the Internet. This service is only listening to localhost and will be tunneled through SSH.
- DNS access (UDP) from the Mail server to the external DNS server at GIAC's ISP (see 2.3 - Name Service).

2.4 Internal network to DMZ

- SSH access to the SSH server on the DMZ from the Internal network for those that are involved with updating the fortunes.
- SSH access to the Mail server for reading e-mail. POP3 is tunneled through SSH.
- SMTP access to the Mail server for sending e-mail.
- HTTP access to the GIAC Web server on the DMZ.

2.5 DMZ to Internal Network

No connections can be made this way!

2.6 Internal network to Internet

The internal IP addresses are NAT'ed by the Firewall using hide NAT.

- Web browsing related traffic: HTTP and HTTPS. All other traffic (e.g. FTP, POP3, Quake etc.) is NOT allowed.

- DNS (UDP) to the external DNS server at GIAC's ISP (see 2.3 - Name Service).
- FTP access (only Passive!) for the System Administrator machine to download software updates, patches etc.
- Telnet access for the System Administrator machine to the internal interface of the Border router.

2.7 Internet to Internal Network

- VPN access to the Fortune database for translators using IPSec with ESP

2.8 Implementation

This section describes in detail how the Firewall rules are set up using Netfilter/iptables. It can be used as a tutorial on how to build the Firewall script. The entire script can be found in APPENDIX A - Netfilter/iptables script. This script is added to *rc.local* so it is started when Linux is booting.

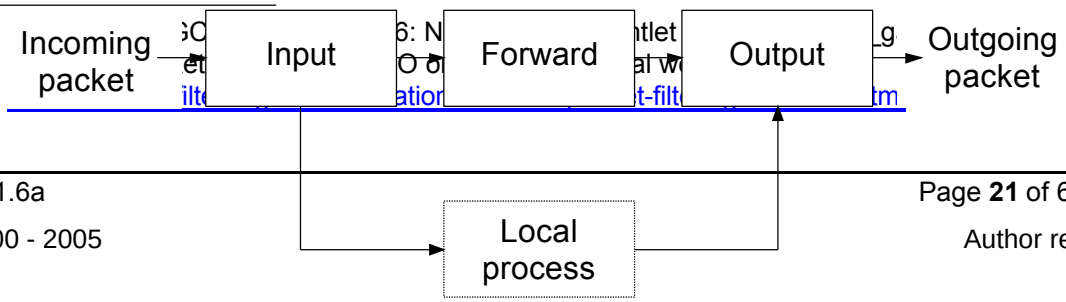
Author's note: I have compiled the Netfilter extensions (e.g. connection tracking, FTP modules etc.) directly into the kernel. Therefore you will not find any Linux modules related commands (depmod, modprobe, insmod etc.) in the script.

The firewall script is built up with the following components:

- Environmental variables
- Kernel settings
- Flushing existing connections
- Chains definition and initialization
- NAT
- Interface rules
- Chains forwarding rules:
  - Internet to DMZ
  - DMZ to Internet
  - Internal network to DMZ
  - DMZ to Internal network
  - Internal network to Internet
  - Internet to Internal network
- Activate chains

All commands in the script are displayed in `courier` font. If a line exceeds the page width, the continuation of it is indented on the next line. Comments in the script are marked with "#".

The Netfilter packet filter uses tables called "chains", which are arranged in the following way[20]:



- Input chain: traffic targeted at firewall (eth0, eth1 or eth2 in the GIAC firewall)
- Output chain: traffic leaving firewall (eth0, eth1 or eth2 in the GIAC firewall)
- Forward chain: traffic crossing firewall
- User chains: can be embedded in first three

Packets would typically be handled by the firewall as follows:

1. When a packet arrives at the network card and it's destined for the firewall itself, the packet passes on to the INPUT chain. If it passes this chain, any local processes waiting for that packet will receive it, or:
2. If forwarding is enabled, and the packet is destined for another network interface, the packet goes right on to the FORWARD chain. If it is accepted, it will be sent to the OUPUT chain.
3. If the OUTPUT chain accepts the packet coming from a local process or the FORWARD chain, this packet continues out to whatever interface it is destined for.

The syntax of the rules for Netfilter/iptables is as follows:

```
iptables <command switches>
-P <chain> <action> = Define chain's policy
-L <chain> = List chain rules
-N <chain> = Create a new user chain
-X <chain> = Delete a user chain
-F <chain> = Flush (delete) all chain rules
-Z <chain> = reset chain counters
-A <chain> = append the following rule
-I # <chain> = insert prior to specified rule
-R # <chain> = replace specified rule
-D # <chain> = delete specified rule
-C <chain> = verify packet handling
```

Rule parameters:

```
-i = receiving interface (eth0, eth1, etc.)
-p = protocol (tcp, udp, 6, 17)
-s = source IP address (address/mask)
-d = destination IP address (address/mask)
```

```
--sport = source port (ssh, http, 135, 8008)
--dport = destination port (ssh, http, 135, 8008)
-j = action to perform (ACCEPT, DROP, REJECT)
-t = table to use (nat, mangle)
! = exception, everything but
```

The rules order is important because Netfilter reads the rules from top to bottom. When a match is made, the firewall will not use further rules down in the list.

2.8.1 Initialization

Initialization of the variables used throughout the script and loading the iptables module.

```
# Location of the iptables binary
IPTABLES="/sbin/iptables"

# The used network interfaces
LOCAL_INT="lo"                # Loopback Interface
INTERNET_INT="eth0"           # External Interface to
Internet
DMZ_INT="eth1"                # DMZ Interface to servers
INTERNAL_INT="eth2"           # Internal Interface to LAN
IPSEC_INT="ipsec0"      # IPsec interface

# Define and automatically obtain the IP-addresses of
     the network interfaces:
INTERNET_IP="`ifconfig $INTERNET_INT | grep \"inet
     addr\" | cut -f 2 -d \":\" | cut -f 1 -d \" \"`"
DMZ_IP="`ifconfig $DMZ_INT | grep \"inet addr\" |
     cut -f 2 -d \":\" | cut -f 1 -d \" \"`"
INTERNAL_IP="`ifconfig $INTERNAL_INT | grep \"inet
     addr\" | cut -f 2 -d \":\" | cut -f 1 -d \" \"`"

# Define the local loop back IP
LOCAL_IP="127.0.0.0"
echo "Done"

# Define GIAC systems
ROUTER="197.22.1.1"
DNSSERVER="ns.giac-isp.net"
WEBSERVER="197.22.1.226"
MAILSERVER="197.22.1.227"
SSHSERVER="197.22.1.228"
GIACSUBNET="192.168.1.0/24"
FORTUNEDB="192.168.1.3"
ADMINMACHINE="192.168.1.4"
PARTNERFW="fw.giac-partner.net"
PARTNERSUBNET="10.1.1.0/24"
```

2.8.2 Kernel Settings

Some kernel settings can be done to help the Firewall doing its work.

```
# Enable forwarding
echo "1" >/proc/sys/net/ipv4/ip_forward

# Enable syn-cookies to prevent syn-flooding attacks
echo "1" >/proc/sys/net/ipv4/tcp_syncookies

# Disable ICMP echo-request to broadcast addresses to
     prevent Smurf attack
echo "1" >/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Disable all ICMP echo-requests to and from the
     firewall
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

# Shut off source-routing and enable IP spoof detection
# This has to be done for all network interfaces
for f in /proc/sys/net/ipv4/conf/*; do
   # Drop all source-routed packets
   echo "0" >$f/accept_source_route

   # Enable source-address verification (anti spoofing).
   # The value 2 means use Ingress filtering as per RFC
     1812.
   echo "2" >$f/rp_filter
done
```

2.8.3 Flushing Existing Connections

When the firewall is re-started, all existing connections have to be removed, to make sure that nothing is left open.

```
$IPTABLES -t filter -F
$IPTABLES -t nat -F
```

2.8.4 Chains Definition and Initialization

For the GIAC firewall the following user chains are defined:

GIAC user chains:

- local: the local loop back address
- internal: internal interface (eth2)
- internet: internet interface (eth0)
- dmz: DMZ interface (eth1)
- internet_to_dmz: forwarding from Internet to DMZ
- dmz_to_internet: forwarding from DMZ to Internet
- internal_to_dmz:: forwarding from internal network to DMZ
- dmz_to_internal: forwarding from DMZ to internal network
- internal_to_internet: forwarding from internal network to Internet

- internet_to_internal: forwarding from Internet to internal network
- vpn_in: forwarding from Internet to internal network over VPN
- vpn_out: forwarding from internal network to Internet over VPN

```
# Default chains definition
$IPTABLES -P INPUT DROP      # Drop all packets to input
$IPTABLES -P FORWARD DROP    # Don't forward anything
$IPTABLES -P OUTPUT ACCEPT   # Accept all packets from
                               output

# Flush the default chains
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
$IPTABLES -F OUTPUT
$IPTABLES -Z INPUT
$IPTABLES -Z FORWARD
$IPTABLES -Z OUTPUT

# User chains definition
$IPTABLES -N local
$IPTABLES -N internal
$IPTABLES -N internet
$IPTABLES -N dmz
$IPTABLES -N internet_to_dmz
$IPTABLES -N dmz_to_internet
$IPTABLES -N internal_to_dmz
$IPTABLES -N dmz_to_internal
$IPTABLES -N internal_to_internet
$IPTABLES -N internet_to_internal
$IPTABLES -N vpn_in
$IPTABLES -N vpn_out

# Flush the GIAC user chains
$IPTABLES -F local
$IPTABLES -F internal
$IPTABLES -F internet
$IPTABLES -F dmz
$IPTABLES -F internet_to_dmz
$IPTABLES -F dmz_to_internet
$IPTABLES -F internal_to_dmz
$IPTABLES -F dmz_to_internal
$IPTABLES -F internal_to_internet
$IPTABLES -F internet_to_internal
$IPTABLES -F vpn_in
$IPTABLES -F vpn_out
```

2.8.5 NAT

Hide NAT is used for the connections from the Internal network to the Internet.

```
# Flush NAT-chain POSTROUTING
$IPTABLES -t nat -F POSTROUTING
# POSTROUTING and SNAT for all packets that go out from
     the Internal network to the Internet
$IPTABLES -t nat -A POSTROUTING -s $GIACSUBNET
     -o $INTERNET_INT -j SNAT --to-source $INTERNET_IP
```

2.8.6 Interface Rules

The rules for each interface of the Firewall: local loop back, Internet, DMZ and Internal.

```
# *** Local loop back ***
# Allow all connections to the local interface, if the
     source IP is the local loop back to make sure that
     Linux can talk to itself
$IPTABLES -A local -m state --state
     NEW,ESTABLISHED,RELATED -i $LOCAL_INT -j ACCEPT


# *** Internal interface ***
# Accept ICMP pings for testing purposes from the
     Internal network
$IPTABLES -A internal -s $GIACSUBNET -p icmp
     --icmp-type echo-request -j ACCEPT
$IPTABLES -A internal -s $GIACSUBNET -p icmp
     --icmp-type echo-reply -j ACCEPT

# Log everything else
$IPTABLES -A internal -j LOG --log-prefix
     "FW-LOG Internal interface:"

#Drop everything else
$IPTABLES -A internal -j DROP


# *** Internet interface ***
# All traffic to the Internet interface is logged
$IPTABLES -A internet -j LOG --log-prefix
     "FW-LOG Internet interface:"

# Allow VPN from the Partner with IKE and ESP
$IPTABLES -A internet -p udp -s $PARTNERFW
     --sport 500 --dport 500 -j ACCEPT
$IPTABLES -A internet -p 50 -j ACCEPT

# Drop everything else. No other connections need to be
     made to the firewall directly from the Internet
$IPTABLES -A internet -j DROP
```

```
# *** DMZ interface ***
# Log everything
$IPTABLES -A dmz -j LOG --log-prefix "FW-LOG DMZ:"
#Drop everything
$IPTABLES -A dmz -j DROP
```

2.8.7 Internet to DMZ

Only traffic to the specific needed services is allowed.

```
# All traffic from the Internet to the DMZ is logged
$IPTABLES -A internet_to_dmz -j LOG --log-prefix
     "FW-LOG Internet to DMZ:"

# Forwarding to the various servers on the DMZ
# Web server HTTP
$IPTABLES -A internet_to_dmz -d $WEBSERVER
     -p tcp --dport http -j ACCEPT

# SSH server
$IPTABLES -A internet_to_dmz -d $SSHSERVER
     -p tcp --dport ssh -j ACCEPT

# Mail server SMTP and SSH (for tunneling POP3)
$IPTABLES -A internet_to_dmz -d $MAILSERVER
     -p tcp --dport smtp -j ACCEPT
$IPTABLES -A internet_to_dmz -d $MAILSERVER
     -p tcp --dport ssh -j ACCEPT

# Stateful check of established connections from the
     mail server
$IPTABLES -A internet_to_dmz -d $MAILSERVER
     -m state --state ESTABLISHED -j ACCEPT

#Drop everything else
$IPTABLES -A internet_to_dmz -j DROP
```

2.8.8 DMZ to Internet

Only the mail server should be able to start a connection out to other mail servers and the external DNS server (on UDP). All other traffic out needs to have an established first (stateful check).

```
# All traffic from the DMZ to the Internet is logged
$IPTABLES -A dmz_to_internet -j LOG --log-prefix "FW-LOG
     DMZ to Internet:"

# Only mail server SMTP and DNS out
$IPTABLES -A dmz_to_internet -s $MAILSERVER
     -p tcp --dport smtp -j ACCEPT
```

```
$IPTABLES -A dmz_to_internet -s $MAILSERVER -d
     $DNSSERVER -p udp --dport 53 -j ACCEPT

# Stateful check of established connections
$IPTABLES -A dmz_to_internet -m state
     --state ESTABLISHED -j ACCEPT
# Drop everything else
$IPTABLES -A dmz_to_internet -j DROP
```

2.8.9 Internal Network to DMZ

Only traffic to the specific needed services is allowed.

```
# SSH server access for fortune update
$IPTABLES -A internal_to_dmz -d $SSHSERVER
     -p tcp --dport ssh -j ACCEPT

# Mail server SMTP and SSH (for tunneling POP3)
$IPTABLES -A internal_to_dmz -d $MAILSERVER
     -p tcp --dport smtp -j ACCEPT
$IPTABLES -A internal_to_dmz -d $MAILSERVER
     -p tcp --dport ssh -j ACCEPT

# Web server HTTP
$IPTABLES -A internal_to_dmz -d $WEBSERVER
     -p tcp --dport http -j ACCEPT

# Log everything else
$IPTABLES -A internal_to_dmz -j LOG --log-prefix
     "FW-LOG Internal to DMZ:"

#Drop everything else
$IPTABLES -A internal_to_dmz -j DROP
```

2.8.10 DMZ to Internal Network

All traffic that is established is allowed (stateful check). Nothing else!

```
# Stateful check of established connections
$IPTABLES -A dmz_to_internal -m state
     --state ESTABLISHED -j ACCEPT

# Log everything else
$IPTABLES -A dmz_to_internal -j LOG --log-prefix
     "FW-LOG DMZ to internal:"

#Drop everything else
$IPTABLES -A dmz_to_internal -j DROP
```

2.8.11 Internal Network to Internet

Only traffic to the specific needed services is allowed.

```
# HTTP and HTTPS out
$IPTABLES -A internal_to_internet
     -p tcp --dport http -j ACCEPT
$IPTABLES -A internal_to_internet
     -p tcp --dport https -j ACCEPT

# DNS out to the external DNS server
$IPTABLES -A internal_to_internet -d $DNSSERVER
     -p udp --dport 53 -j ACCEPT

# FTP out from the administrator machine
$IPTABLES -A internal_to_internet -s $ADMINMACHINE
     -p tcp --dport ftp -j ACCEPT

# Telnet access to the Border router from the
     administrator machine
$IPTABLES -A internal_to_internet -s $ADMINMACHINE
     -d $ROUTER -p tcp --dport telnet -j ACCEPT

# Stateful check of Partner connections to Fortune
     database (no connection initiations are
     necessary/allowed from the Fortune database to the
     partner)
$IPTABLES -A vpn_out -m state
     --state ESTABLISHED,RELATED -j ACCEPT

# Log everything else
$IPTABLES -A internal_to_internet -j LOG --log-prefix
     "FW-LOG Internal to internet:"

# Drop everything else.
$IPTABLES -A internal_to_internet -j DROP
```

2.8.12 Internet to Internal network

VPN access to the Fortune database for translator partners and established traffic is allowed (stateful check)

```
# Packets from the Internet to Internal network must
     have an establishment (stateful check).
$IPTABLES -A internet_to_internal -m state
     --state ESTABLISHED,RELATED -j ACCEPT

# Log everything else
$IPTABLES -A internet_to_internal -j LOG --log-prefix
     "FW-LOG Internet to internal:"

# VPN from the Partner to Fortune DB
$IPTABLES -A vpn_in -s $PARTNERSUBNET -d $FORTUNEDB
     -j ACCEPT
```

```
# Drop everything else.
$IPTABLES -A internet_to_internal -j DROP
```

2.8.13 Activate chains

Now all the defined user chains have to be activated.

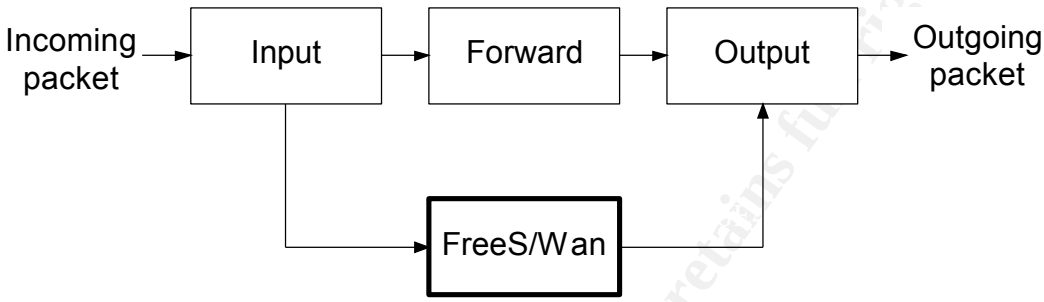
```
# Activate the user chains
$IPTABLES -A INPUT -i $LOCAL_INT  -j local
$IPTABLES -A INPUT -i $INTERNAL_INT -j internal
$IPTABLES -A INPUT -i $INTERNET_INT -j internet
$IPTABLES -A INPUT -i $DMZ_INT -j dmz
$IPTABLES -A FORWARD -i $INTERNET_INT -o $DMZ_INT
     -j internet_to_dmz
$IPTABLES -A FORWARD -i $DMZ_INT -o $INTERNET_INT
     -j dmz_to_internet
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $DMZ_INT
     -j internal_to_dmz
$IPTABLES -A FORWARD -i $DMZ_INT -o $INTERNAL_INT
     -j dmz_to_internal
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $INTERNET_INT
     -j internal_to_internet
$IPTABLES -A FORWARD -i $INTERNET_INT -o $INTERNAL_INT
     -j internet_to_internal
$IPTABLES -A FORWARD -i $IPSEC_INT -o $INTERNAL_INT
     -j vpn_in
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $IPSEC_INT
     -j vpn_out
```

3 VPN

The VPN software that is used is Frees/Wan 1.95. The installation instructions in the next sections are taken from Frees/Wan's web site[21].

FreeS/Wan uses ESP with 3DES encryption by default.

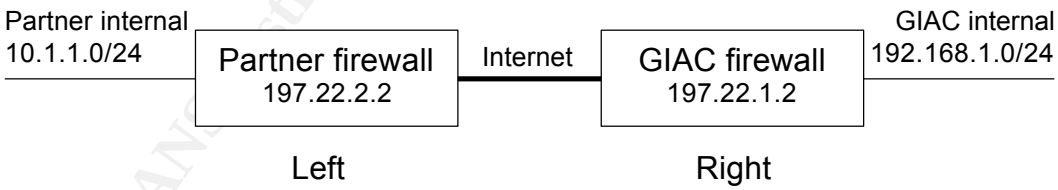
FreeS/Wan is integrated into the Netfilter chains (see 2.8 - Implementation) in the following way:



FreeS/Wan is running as a local process and has its own "network interface" called *ipsec[number]* (ipsec0 in GIAC's firewall). This interface is treated in the same way as the *eth[number]* interfaces by Netfilter, i.e. using INPUT, OUTPUT and FORWARD chains.

3.1 Network overview

The VPN is set up between the GIAC translator partner's firewall (fw.giac-partner.net) and GIAC's firewall. Of course more VPNs can be set up in the same way.
The network between GIAC and the partner can be represented as follows:



FreeS/Wan uses the naming conventions "Left" and "Right". The shown settings above are used in the configuration files (see 3.3 - Configuration files).

[21] See: Freeswan's Configuration documentation:
http://www.freeswan.org/freeswan_trees/freeswan-1.95/doc/config.html

3.2 RSA set up

3.2.1 Key generation

Before the VPN is set up, RSA public and private keys have to be generated. RSA keys are only used for authentication not encryption. Running the following command on each firewall generates the private RSA keys:

```
ipsec newhostkey > /etc/ipsec.secrets
```

Set security on this private key(!):

```
chmod 600 /etc/ipsec.secrets
```

3.2.2 Public key creation and distribution

The public keys for each host are by running the following commands on each firewall:

On the partner firewall:
```
ipsec showhostkey --left > /etc/leftrsakey
```

On GIAC's firewall:
```
ipsec showhostkey --right > /etc/rightrsakey
```

The keys will be put into the configuration file for the firewalls as described in the next section.

3.3 Configuration files

The configuration file for FreeS/Wan is called */etc/ipsec.conf*
By default the format of the file looks like this:

```
# Basic configuration
config setup
       # %defaultroute is okay for most simple cases.
       interfaces=%defaultroute
       # Debug-logging controls
       klipsdebug=none
       plutodebug=none
       # Use auto= parameters in conn descriptions to
       # control startup actions.
       plutoload=%search
       plutostart=%search
       # Close down old connection when new one using
       # same ID shows up.
       uniqueids=yes

# defaults for subsequent connection descriptions
# (mostly to fix internal defaults which, in retrospect,
```

```
# were badly chosen)
conn %default
       keyingtries=0
       disablearrivalcheck=no
       authby=rsasig
       leftrsasigkey=%dns
       rightrsasigkey=%dns

# sample VPN connection
conn sample
# Left security gateway, subnet behind it, next hop
# toward right.
       left=10.0.0.1
       leftsubnet=172.16.0.0/24
       leftnexthop=10.22.33.44
       # Right security gateway, subnet behind it, next
       # hop toward left.
       right=10.12.12.1
       rightsubnet=192.168.0.0/24
       rightnexthop=10.101.102.103
       # To authorize this connection, but not actually
       # start it, at startup, uncomment this.
       #auto=add
```

For GIAC the config file looks as follows:
(The RSA public keys are not completely shown for better readability).

```
config setup
       interfaces=%defaultroute
       klipsdebug=all
       plutodebug=all
       plutoload=%search
       plutostart=%search
       uniqueids=yes

conn %default
       keyingtries=0
       authby=rsasig

conn giac
       leftid=@partner.fw.giac-partner.net
       leftrsasigkey=0sAQO+OiRsBxb1EC...
       left=197.22.2.2
       leftsubnet=10.1.1.0/24
       rightid=@giac.197.22.1.2
       rightrsasigkey=0sAQOuukCtRZs/...
       right=197.22.1.2
       rightsubnet=192.168.1.0/24
       auto=add
```

3.4 Starting and Testing

The VPN connection is started when the ipsec daemon in Linux is started or it can be started manually:

```
ipsec auto --add giac
ipsec auto --up giac
```

This will cause the connection to start up:

```
104 "giac" #5: STATE_MAIN_I1: initiate
106 "giac" #5: STATE_MAIN_I2: sent MI2, expecting MR2
108 "giac" #5: STATE_MAIN_I3: sent MI3, expecting MR3
004 "giac" #5: STATE_MAIN_I4: ISAKMP SA established
112 "giac" #6: STATE_QUICK_I1: initiate
004 "giac" #6: STATE_QUICK_I2: sent QI2, IPsec SA
     established
```

Tcpdump shows the following:

```
tcpdump -i eth0 -p -vvv not arp -t

197.22.1.2.500 > fw.giac-partner.net.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>0000000000000000: phase 1 I ident: [|sa] (DF) (ttl 64,
id 0, len 204)

fw.giac-partner.net.500 > 197.22.1.2.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 1 R ident: [|sa] (DF) (ttl 64,
id 0, len 108)

197.22.1.2.500 > fw.giac-partner.net.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 1 I ident: [|ke] (DF) (ttl 64,
id 0, len 272)

fw.giac-partner.net.500 > 197.22.1.2.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 1 R ident: [|ke] (DF) (ttl 64,
id 0, len 272)

197.22.1.2.500 > fw.giac-partner.net.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 1 I ident[E]: [|id] (DF) (ttl
64, id 0, len 344)
fw.giac-partner.net.500 > 197.22.1.2.500: isakmp 1.0
msgid 00000000 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 1 R ident[E]: [|id] (DF) (ttl
64, id 0, len 344)
```

```
197.22.1.2.500 > fw.giac-partner.net.500: isakmp 1.0
msgid 0f675655 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 2/others I oakley-quick[E]:
[|hash] (DF) (ttl 64, id 0, len 408)

fw.giac-partner.net.500 > 197.22.1.2.500: isakmp 1.0
msgid 0f675655 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 2/others R oakley-quick[E]:
[|hash] (DF) (ttl 64, id 0, len 376)

197.22.1.2.500 > fw.giac-partner.net.500:  [udp sum
ok]isakmp 1.0 msgid 0f675655 cookie faf33f7b3b4b3f1d-
>7a927b61301923b3: phase 2/others I oakley-quick[E]:
[|hash] (DF) (ttl 64, id 0, len 80)
```

Tcpdump sniffing on the input of GIAC's firewall displays traffic between a host on the partner's subnet to the Fortune database as follows:

```
tcpdump -i eth0 -vvv -p not arp

11:21:54.971427 fw.giac-partner.net > 197.22.1.2:
ESP(spi=0x7ef62ffd,seq=0x27f) (ttl 64, id 7688, len 104)

11:21:54.971782 fw.giac-partner.net > 197.22.1.2:
ESP(spi=0x7ef62ffd,seq=0x280) (ttl 64, id 7689, len 104)

11:21:54.973008 197.22.1.2 > fw.giac-partner.net:
ESP(spi=0xd69b9424,seq=0x20f) (ttl 64, id 19301, len
104)

11:21:54.973428 197.22.1.2 > fw.giac-partner.net:
ESP(spi=0xd69b9424,seq=0x210) (ttl 64, id 19302, len
104)

11:21:54.975577 fw.giac-partner.net > 197.22.1.2:
ESP(spi=0x7ef62ffd,seq=0x281) (ttl 64, id 7690, len 96)

11:21:54.975991 fw.giac-partner.net > 197.22.1.2:
ESP(spi=0x7ef62ffd,seq=0x282) (ttl 64, id 7691, len 232)

11:21:54.977498 197.22.1.2 > fw.giac-partner.net:
ESP(spi=0xd69b9424,seq=0x211) (ttl 64, id 19303, len
184)
```

The original unencrypted NetBIOS traffic sniffed on the Inside interface of the GIAC firewall is:

```
tcpdump -i eth2 -p -vvv not arp -n
```

```
11:21:54.972315 10.1.1.2.1030 > 192.168.1.3.445: S [tcp
sum ok] 4197936269:4197936269(0) win 16384 <mss
1460,nop,nop,sackOK> (DF) (ttl 126, id 703, len 48)

11:21:54.972336 10.1.1.2.1031 > 192.168.1.3.139: S [tcp
sum ok] 4197984491:4197984491(0) win 16384 <mss
1460,nop,nop,sackOK> (DF) (ttl 126, id 704, len 48)

11:21:54.972546 192.168.1.3.445 > 10.1.1.2.1030: S [tcp
sum ok] 1100691142:1100691142(0) ack 4197936270 win
17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 3909,
len 48)

11:21:54.972549 192.168.1.3.139 > 10.1.1.2.1031: S [tcp
sum ok] 1100753596:1100753596(0) ack 4197984492 win
17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 3910,
len 48)

11:21:54.976476 10.1.1.2.1030 > 192.168.1.3.445: . [tcp
sum ok] 1:1(0) ack 1 win 17520 (DF) (ttl 126, id 705,
len 40)

11:21:54.976490 10.1.1.2.1030 > 192.168.1.3.445: P
1:138(137) ack 1 win 17520 (DF) (ttl 126, id 708, len
177)

11:21:54.977008 192.168.1.3.445 > 10.1.1.2.1030: P
1:90(89) ack 138 win 17383 (DF) (ttl 128, id 3911, len
129)

11:21:54.982823 10.1.1.2.1030 > 192.168.1.3.445: P
138:326(188) ack 90 win 17431 (DF) (ttl 126, id 709, len
228)
```

4 Testing ACL's

This section describes how 3 example firewall ACL's can be tested.

4.1 Contacting web server

The rules that are relevant here are:

- Internet to DMZ rule (see 2.8.7)
  ```
  $IPTABLES -A internet_to_dmz -d $WEBSERVER
     -p tcp --dport http -j ACCEPT
  ```

- DMZ to Internet rule (see 2.8.8)
  ```
  $IPTABLES -A dmz_to_internet -m state
    --state ESTABLISHED -j ACCEPT
  ```

An easy way to test this, is to open a telnet connection to port 80:

```
telnet 197.22.1.226
HEAD / HTTP/1.0
```

If a legal HTTP reply is received, the rule is working. Even easier testing can be done by opening GIAC's web site with a browser.

4.2 Surfing on the Internet

The following rules open up for web browsing from the internal network:

- Internet network to Internet (see 2.8.11)

```
$IPTABLES -A internal_to_internet
 -p tcp --dport http -j ACCEPT
$IPTABLES -A internal_to_internet
 -p tcp --dport https -j ACCEPT
$IPTABLES -A internal_to_internet -d $DNSSERVER
 -p udp --dport 53 -j ACCEPT
```

- Internet to Internal network (see 2.8.12)

```
$IPTABLES -A internet_to_internal -m state
 --state ESTABLISHED,RELATED -j ACCEPT
```

To test this, just open a HTTP and HTTPS web site on a browser on the internal network using the DNS name of the web sites.

4.3 Sending mails from the mail server

The following rules apply here:

- DMZ to Internet (see 2.8.8)

```
$IPTABLES -A dmz_to_internet -s $MAILSERVER
 -p tcp --dport smtp -j ACCEPT
$IPTABLES -A dmz_to_internet -s $MAILSERVER -d
 $DNSSERVER -p udp --dport 53 -j ACCEPT
```

- Internet to DMZ (see 2.8.7)

```
$IPTABLES -A internet_to_dmz -d $MAILSERVER
 -m state --state ESTABLISHED -j ACCEPT
```

Try to telnet an external mail server from GIAC's mail server:

```
telnet <some mail server> 25
```

If this is allowed and SMTP commands can be executed, then this part is working.
Now contact the DNS server from the GIAC mail server

```
nslookup
server ns.giac-isp.net
set type = mx
<some domain name>
```

If a reply is received, the rules are working.

5 Linux Servers

All the Linux Servers (including the Firewall!) are hardened before they are put into production. SecurityFocus.com has an excellent article on Linux Kernel Hardening[22].
The description of kernel hardening would go beyond the scope of this document.

[22] See: The SecurityFocus web site - Linux Kernel Hardening by Anton Chuvakin, Ph.D.: http://www.securityfocus.com/infocus/1539

Assignment 3 - Audit Security Architecture

1 Plan

This section describes the planning and execution of the security audit of GIAC's firewall.

1.1 Inform

At least one week before starting the test, all involved personnel have to be informed about the test and the time frame of the test. These would typically be the IT manager, system administrators and users (GIAC internal & partners).
IT personnel need to be warned about the fact that IDS systems might (should) set off a lot alarms and log files grow considerably during the test. There might be a Denial-of-Service (DoS) risk, which the IT personnel should be aware of.
Get hold of the names of the contact persons and their backups that can restart the firewall and other systems immediately if a DoS occurs. Make sure that these contact persons are available during the entire test period. Call these contacts persons 5 minutes before the test is started. If they're not there, the test is postponed.

1.2 Test execution

The test of GIAC's firewall consists of:

- Network scan: ICMP and TCP ping scanning of GIAC's external address space.
- Information gathering: DNS and Whois lookup of the found IP addresses. Traceroute of these IP addresses.
- Port Scanning: TCP and UDP port scan of GIAC's firewall.
- Fingerprinting: TCP/IP fingerprinting to identify the OS of the firewall.
- Penetration tests: Attempt to send illegal packets through the firewall.
- ICMP tests: Sending different types of ICMP packets to and through the firewall.

1.3 Time frame

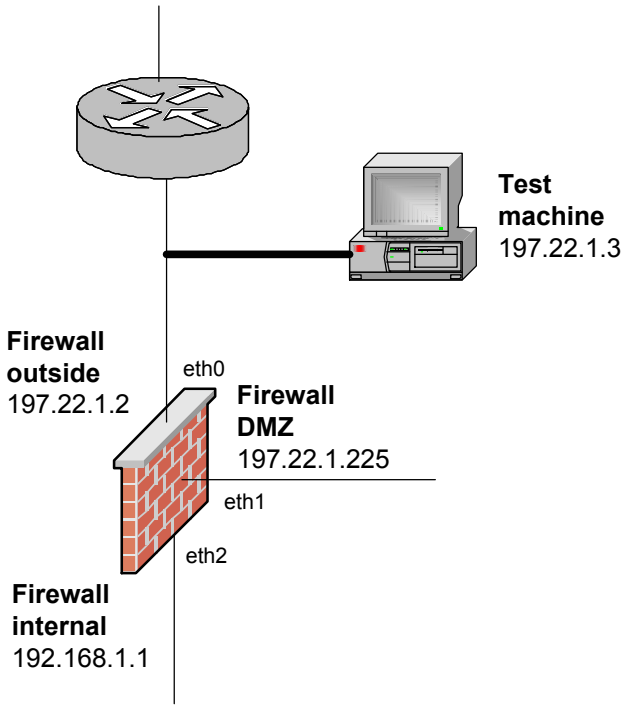
The test itself will take 1 working day, 8 hours.
Reporting and QA of the results will take 2 hours.
The test will take place during normal office hours, because it would be rather expensive to conduct the test outside office hours (1 tester plus at least 1 contact person have to work after hours at double rate). This choice is made because the impact of the test is expected to be low.

1.4 Cost

Since the test is done during office hours, only the tester works full-time on it, not the contact person(s). At a rate of $250 per hour, the test will cost app. $2500.
The time it takes to correct the problems that might be found during the test are not taken into consideration here.

2 Test Execution and Results

This section describes the execution of a security audit of GIAC's firewall and the results from these tests.
The test is run directly on the firewall, without any routers or other systems in between, as shown in the picture below:



The test machine's operating system is Linux RedHat 7.2 - kernel 2.4.9-21. The following default route is added to the test machine to make sure that the traffic goes the correct direction:

```
route add default gw 197.22.1.2
```

Other default routes are removed.

2.1 Network scan

The network scan tries to identify the systems that are present in GIAC's public address space.

2.1.1 ICMP ping sweep

Using ICMP echo requests is the simplest way to check system availability.

Nmap[23] has an option that can do this. We use version 2.53. The command that has to be run is:

```
nmap -sP -PI 197.22.1.0/24
```

This will only test for ICMP echo requests/replies.

The result of this command is:

```
Starting nmap V. 2.53 by fyodor@insecure.org
( www.insecure.org/nmap/ )
Host  (197.22.1.1) appears to be up.
Nmap run completed -- 256 IP addresses (1 hosts up)
scanned in 5 seconds
```

This is the result we expected, as we block all ICMP traffic at the firewall. Only the router replies, which is acceptable.

2.1.2 TCP ping sweep

Nmap can also do a "TCP ping" by sending SYN or ACK packets to a specified port. If a host in the range replies with SYN/ACK or RST for this port, it is assumed to be up.
With this test we expect to find the web, mail and ssh server, because these have TCP ports open. The router has port 23 (telnet) open, but this is not the reason it should be found, because it only listens to the firewall's IP address. We should find the router because it replies with RST for closed TCP ports. All other TCP ports do not reply due to our firewall rules.

The TCP ping with ACK packets command is:

```
nmap -sP -PT<portnumber> 197.22.1.0/24
```

With the results for selected TCP ports:

```
TCP probe port is 22
Host  (197.22.1.1) appears to be up.
Host  (197.22.1.228) appears to be up.
Nmap run completed -- 256 IP addresses (2 hosts up)
scanned in 5 seconds

TCP probe port is 23
Host  (197.22.1.1) appears to be up.
Nmap run completed -- 256 IP addresses (1 hosts up)
scanned in 5 seconds

TCP probe port is 25
Host  (197.22.1.1) appears to be up.
Host  (197.22.1.227) appears to be up.
Nmap run completed -- 256 IP addresses (2 hosts up)
```

[23] See: The Insecure.org web site: http://www.insecure.org/nmap

```
scanned in 5 seconds

TCP probe port is 80
Host  (197.22.1.1) appears to be up.
Host  (197.22.1.226) appears to be up.
Nmap run completed -- 256 IP addresses (2 hosts up)
scanned in 5 seconds
```

The result of the test is the same when using SYN packets.

2.2 Information gathering

2.2.1 Whois

Running the next command on the test machine retrieves Whois information:

```
whois 197.22.1.0@whois.ripe.net
```

The result is:

```
[whois.ripe.net]
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more
   information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-
   services/db/copyright.html

inetnum:      0.0.0.0 - 255.255.255.255
netname:      IANA-BLK
descr:        The whole IPv4 address space
country:      NL
admin-c:      IANA1-RIPE
tech-c:       IANA1-RIPE
status:       ALLOCATED UNSPECIFIED
remarks:      The country is really worldwide.
remarks:      This address space is assigned at various
              other places in
remarks:      the world and might therefore not be in
              the RIPE database.
mnt-by:       RIPE-NCC-HM-MNT
mnt-lower:    RIPE-NCC-HM-MNT
mnt-routes:   RIPE-NCC-NONE-MNT
changed:      bitbucket@ripe.net 20010529
source:       RIPE

role:         Internet Assigned Numbers Authority
address:      see http://www.iana.org.
```

```
e-mail:       bitbucket@ripe.net
admin-c:      IANA1-RIPE
tech-c:       IANA1-RIPE
nic-hdl:      IANA1-RIPE
remarks:      For more information on IANA services
remarks:      go to IANA web site at
http://www.iana.org.
mnt-by:       RIPE-NCC-MNT
changed:      bitbucket@ripe.net 20010411
source:       RIPE
```

Author's note: Since we use a reserved address space, no "real" answer is retrieved.

2.2.2 DNS lookup

Using the Linux *dig* command we can do a DNS lookup of the GIAC IP-addresses:

```
dig @<dnsserver> <domainname>
```
or
```
dig @<dnsserver> -x <ip address>
```

Author's note: The DNS names we use are fictitious, so this test cannot actually be done.

2.2.3 Traceroute

Traceroute can be done using ICMP, UDP and TCP. The latter is expected to be the most successful in our set up. ICMP and UDP traceroute should only be possible on the router, since our firewall ought to block it for other IP addresses.

For ICMP traceroute we use the default Linux *traceroute* tool:

```
traceroute -I <target>
```

Result:

```
traceroute -I 197.22.1.1
traceroute to 197.22.1.1 (197.22.1.1), 30 hops max, 38
byte packets
1  197.22.1.1 (197.22.1.1) 1.754 ms  0.839 ms  0.801 ms

traceroute -I 197.22.1.2
traceroute to 197.22.1.2 (197.22.1.2), 30 hops max, 38
byte packets
1  * * *

traceroute -I 197.22.1.225
traceroute to 197.22.1.225 (197.22.1.225), 30 hops max,
```

```
38 byte packets
1  * * *

traceroute -I 197.22.1.226
traceroute to 197.22.1.226 (197.22.1.226), 30 hops max,
38 byte packets
1  197.22.1.225 (197.22.1.225) 0.700 ms  0.661 ms  0.543
ms
2  * * *
```

The results for the other hosts are the same. We can actually see the firewall in the trace to these hosts. The reason is that the firewall sends *time exceeded in-transit* ICMP messages back to our test machine. We might consider adding a rule to the firewall that eliminates these packets, because only routers should send these.

For UDP tracerouting the same tool is used (it uses UDP by default):

```
traceroute <target>
```

The results are exactly the same as the ones from the ICMP traceroute.

TCP tracerouting is done with the help of Michael Toren's tool *tcptraceroute*[24].

```
tcptraceroute <target> [port]
```

This tool also uses increasing TTL settings to trace the route, but instead of using ICMP echo requests or UDP packets, it sends TCP SYN packets to a specified port (default port 80). If the tracerouted host replies with a SYN/ACK or RST, it can be tracerouted. See the results below.

```
tcptraceroute 197.22.1.1
Selected device eth0, address 197.22.1.3, port 38450 for
outgoing packets
Tracing the path to 197.22.1.1 on TCP port 80, 30 hops
max
1  197.22.1.1 (197.22.1.1) [closed] 1.134 ms  0.995 ms
0.972 ms

tcptraceroute 197.22.1.2
Selected device eth0, address 197.22.1.3, port 38451 for
outgoing packets
Tracing the path to 197.22.1.2 on TCP port 80, 30 hops
max
1  * * *

tcptraceroute 197.22.1.225
```

[24] See: Michael Toren's web site: http://michael.toren.net/code/tcptraceroute

```
Selected device eth0, address 197.22.1.3, port 38452 for
outgoing packets
Tracing the path to 197.22.1.225 on TCP port 80, 30 hops
max
1  * * *

tcptraceroute 197.22.1.226
Selected device eth0, address 197.22.1.3, port 38453 for
outgoing packets
Tracing the path to 197.22.1.226 on TCP port 80, 30 hops
max
1  197.22.1.2 (197.22.1.2) 1.284 ms  0.975 ms  0.862 ms
2  197.22.1.226 (197.22.1.226) [open]  1.936 ms  1.115
ms  1.078 ms

tcptraceroute 197.22.1.227 25
Selected device eth0, address 197.22.1.3, port 38454 for
outgoing packets
Tracing the path to 197.22.1.227 on TCP port 25, 30 hops
max
1  197.22.1.2 (197.22.1.2) 1.284 ms  0.975 ms  0.862 ms
2  197.22.1.227 (197.22.1.227) [open]  1.139 ms  1.111
ms  1.068 ms

tcptraceroute 197.22.1.228 22
Selected device eth0, address 197.22.1.3, port 38455 for
outgoing packets
Tracing the path to 197.22.1.228 on TCP port 22, 30 hops
max
1  197.22.1.2 (197.22.1.2) 1.284 ms  0.975 ms  0.862 ms
2  197.22.1.228 (197.22.1.228) [open]  1.161 ms  1.734
ms  1.065 ms
```

Once again, we might consider adding a rule to the firewall that eliminates TTL exceeded packets.

2.3 Port Scanning

2.3.1 TCP ports

First we scan our IP addresses for TCP ports 1-65535 using nmap:

```
nmap -P0 -p 1-65535 <target>
```

We use the -P0 option because the GIAC hosts do not reply to ping.

Results:

```
nmap -P0 -p 1-65535 197.22.1.2
All 65535 scanned ports on  (197.22.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
```

```
in 66231 seconds

nmap -P0 -p 1-65535 197.22.1.226
Interesting ports on  (197.22.1.226):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
80/tcp     open        http
Nmap run completed -- 1 IP address (1 host up) scanned
in 13408 seconds

nmap -P0 -p 1-65535 197.22.1.227
Interesting ports on  (197.22.1.227):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
25/tcp     open        smtp
Nmap run completed -- 1 IP address (1 host up) scanned
in 14318 seconds

nmap -P0 -p 1-65535 197.22.1.228
Interesting ports on  (197.22.1.228):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
22/tcp     open        ssh
Nmap run completed -- 1 IP address (1 host up) scanned
in 13792 seconds
```

Then we do the same scan, but now we use TCP source ports 20 and 53, because in some cases these ports can bypass some ACL's (this should not be the case with our firewall though).

```
nmap -P0 -sS -g 20 -p 1-65535 <target>
```

We need to use the -sS (raw SYN scanning) option, because nmap cannot do its default connect() scan with the -g option set. Nmap's warning:

```
WARNING:  -g is incompatible with the default connect()
scan (-sT).  Use a raw scan such as -sS if you want to
set the source port.
```

Results:

```
nmap -P0 -sS -g20 -p 1-65535 197.22.1.2
All 65535 scanned ports on  (197.22.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 32293 seconds

nmap -P0 -sS -g20 -p 1-65535 197.22.1.226
Interesting ports on  (197.22.1.226):
```

```
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
80/tcp     open        http
Nmap run completed -- 1 IP address (1 host up) scanned
in 11052 seconds

nmap -P0 -sS -g20 -p 1-65535 197.22.1.227
Interesting ports on  (197.22.1.227):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
25/tcp     open        smtp
Nmap run completed -- 1 IP address (1 host up) scanned
in 12143 seconds

nmap -P0 -sS -g20 -p 1-65535 197.22.1.228
Interesting ports on  (197.22.1.228):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
22/tcp     open        ssh
Nmap run completed -- 1 IP address (1 host up) scanned
in 11918 seconds

nmap -P0 -sS -g53 -p 1-65535 197.22.1.2
All 65535 scanned ports on  (197.22.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 31304 seconds

nmap -P0 -sS -g53 -p 1-65535 197.22.1.226
Interesting ports on  (197.22.1.226):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
80/tcp     open        http
Nmap run completed -- 1 IP address (1 host up) scanned
in 11165 seconds

nmap -P0 -sS -g53 -p 1-65535 197.22.1.227
Interesting ports on  (197.22.1.227):
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
25/tcp     open        smtp
Nmap run completed -- 1 IP address (1 host up) scanned
in 11235 seconds

nmap -P0 -sS -g53 -p 1-65535 197.22.1.228
Interesting ports on  (197.22.1.228):
```

```
(The 65534 ports scanned but not shown below are in
state: filtered)
Port       State       Service
22/tcp     open        ssh
Nmap run completed -- 1 IP address (1 host up) scanned
in 11822 seconds
```

These are the results we expected. No unexpected services are open. Nmap displays the closed ports as *filtered*. It does this because closed ports on the GIAC hosts do not reply at all to TCP SYN requests.

2.3.2 UDP ports

Also for UDP we perform a full regular scan and a source port scan for port 53, because this port might bypass some ACL's (should not happen with our firewall)

```
nmap -P0 -sU -p 1-65535 <target>
```

Results:

```
nmap -P0 -sU -p 1-65535 197.22.1.2
All 65535 scanned ports on  (197.22.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 75035 seconds

nmap -P0 -sU -p 1-65535 197.22.1.226
All 65535 scanned ports on  (197.22.1.226) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 64371 seconds

nmap -P0 -sU -p 1-65535 197.22.1.227
All 65535 scanned ports on  (197.22.1.227) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 63245 seconds

nmap -P0 -sU -p 1-65535 197.22.1.228
All 65535 scanned ports on  (197.22.1.228) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 62312 seconds

nmap -P0 -sU -g 53 -p 1-65535 197.22.1.2
All 65535 scanned ports on  (197.22.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 74498 seconds

nmap -P0 -sU -g 53 -p 1-65535 197.22.1.226
All 65535 scanned ports on  (197.22.1.226) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 53277 seconds
```

```
nmap -P0 -sU -g 53 -p 1-65535 197.22.1.227
All 65535 scanned ports on  (197.22.1.227) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 54322 seconds

nmap -P0 -sU -g 53 -p 1-65535 197.22.1.228
All 65535 scanned ports on  (197.22.1.228) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned
in 53629 seconds
```

2.4 Fingerprinting

Fingerprinting is done also by using nmap. Nmap gets the best fingerprinting results with at least one open and one closed TCP port. This means that we don't expect any results on the firewall, but we might get something from our hosts on the DMZ.

```
nmap -P0 -O -p <open-closed port> <target>
```

Results:

```
nmap -P0 -O 197.22.1.2
Interesting ports on  (197.22.1.2):
Port       State       Service
80/tcp     open        http
81/tcp     filtered    unknown

TCP Sequence Prediction: Class=random positive
increments
                         Difficulty=1637737 (Good luck!)
No OS matches for host

nmap -P0 -O -p 80-81 197.22.1.226
Interesting ports on  (197.22.1.226):
Port       State       Service
80/tcp     open        http
81/tcp     filtered    unknown

TCP Sequence Prediction: Class=random positive
increments
                         Difficulty=1637737 (Good luck!)
No OS matches for host

nmap -P0 -O -p 25-26 197.22.1.227
Interesting ports on  (197.22.1.227):
Port       State       Service
25/tcp     open        smtp
26/tcp     filtered    unknown

TCP Sequence Prediction: Class=random positive
increments
                         Difficulty=1751081 (Good luck!)
No OS matches for host

nmap -P0 -O -p 22-23 197.22.1.228
Interesting ports on  (197.22.1.228):
Port       State       Service
22/tcp     open        ssh
23/tcp     filtered    telnet
```

```
TCP Sequence Prediction: Class=random positive
increments
                         Difficulty=2682383 (Good luck!)
No OS matches for host
```

No operating system fingerprints could be made with nmap, which is good. Another plus is that TCP sequence prediction is virtually impossible.

2.5 ICMP tests

ICMP tests are done with the *icmpush* tool by Slayer[25]. This tool can send various types of ICMP messages. The one we are most interested in, are timestamp and netmask requests. If one or more of the GIAC hosts reply, they are giving away too much information. These types of requests should never be granted. The requests are send in the following way:

```
icmpush -tstamp <target>
icmpush -mask <target>
```

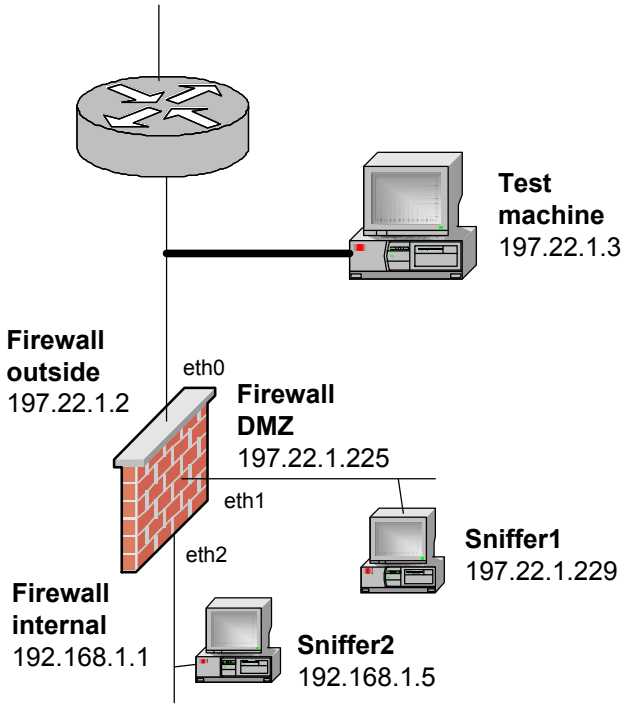
Results:
None of the GIAC hosts replied to either a timestamp or netmask request.

2.6 Penetration tests

The penetration tests are done with the help of a "sniffer" on both the DMZ and the Internal network. These sniffers are just plain Linux machines running tcpdump in promiscuous mode.
We will try to send normal and spoofed TCP, UDP and ICMP packets from one subnet to another using our test machine and the sniffers, while we are listening with tcpdump on the another subnet.

The sending machine will have the firewall as the default gateway to make sure that all packets are sent directly to the firewall.

[25] Downloaded from Packetstorm Security: http://www.packetstormsecurity.org/UNIX/scanners/



2.6.1 Internet to DMZ

First we will start a tcpdump on sniffer1:

```
tcpdump -i eth0 -n host 197.22.1.3
```

In this way no DNS lookups are done (-n option), which speeds up tcpdump and we will only see traffic from our test machine.

Our first test is to try sending TCP, UDP and ICMP packets from 197.22.1.3 to 197.22.1.229. The easiest way to do this is using an nmap and icmpush:

```
nmap -P0 -p 1-65535 197.22.1.229
nmap -P0 -sU -p 1-65535 197.22.1.229
icmpush -echo 197.22.1.229
```

None of the packets got to sniffer1, like we expected.

Then we do the same tests, but now we spoof the source address using the DMZ IP address of the firewall. The firewall ought to drop these packets, but in some firewalls the routing mechanism gets confused and the packets are delivered to the spoofed interface, which might forward them.

We should also remember to make tcpdump on sniffer1 listen to the spoofed address and not the test machine's IP address:

```
tcpdump -i eth0 -n host 197.22.1.225

nmap -P0 -e eth0 -sS -S 197.227.1.225
```

```
      -p 1-65535 197.22.1.229
nmap -P0 -e eth0 -sU -S 197.227.1.225
      -p 1-65535 197.22.1.229
icmpush -echo -sp 197.22.1.225 197.22.1.229
```

We use the -sS and -e options because otherwise nmap has problems spoofing the source address (-S option).

None of the spoofed packets reached sniffer1, which is expected behavior.

2.6.2 DMZ to Internet

On our testmachine we start a tcpdump:

```
tcpdump -i eth0 -n host 197.22.1.229
```

and for the spoof tests:

```
tcpdump -i eth0 -n host 197.22.1.2
```

Then we run the tests from sniffer1:

```
nmap -P0 -p 1-65535 197.22.1.3
nmap -P0 -sU -p 1-65535 197.22.1.3
icmpush -echo 197.22.1.3
```

None of the packets got to the test machine.

```
nmap -P0 -e eth0 -sS -S 197.227.1.2
      -p 1-65535 197.22.1.3
nmap -P0 -e eth0 -sU -S 197.227.1.2
      -p 1-65535 197.22.1.3
icmpush -echo -sp 197.22.1.2 197.22.1.3
```

Once again, no results on the test machine.

2.6.3 Internet to Internal network

On sniffer2 we start a tcpdump:

```
tcpdump -i eth0 -n host 197.22.1.3
```

and for the spoof tests:

```
tcpdump -i eth0 -n host 192.168.1.1
```

Then we run the tests from the test machine:

```
nmap -P0 -p 1-65535 192.168.1.5
nmap -P0 -sU -p 1-65535 192.168.1.5
icmpush -echo 192.168.1.5
```

```
nmap -P0 -e eth0 -sS -S 192.168.1.1
     -p 1-65535 192.168.1.5
nmap -P0 -e eth0 -sU -S 192.168.1.1
     -p 1-65535 192.168.1.5
icmpush -echo -sp 192.168.1.1 192.168.1.5
```

Fortunately, none of the packets arrived at sniffer2.

2.6.4 Internal network to Internet

On the test machine we start a tcpdump:

```
tcpdump -i eth0 -n host 197.22.1.2
```

For the spoof tests this is the same.

We need to snif the traffic from the external interface of the firewall, because the internal address of sniffer2 is NAT'ed.

Then we run the tests from sniffer2:

```
nmap -P0 -p 1-65535 197.22.1.3
nmap -P0 -sU -p 1-65535 197.22.1.3
icmpush -echo 197.22.1.3
```

The output of tcpdump on the test machine looks as follows:

```
197.22.1.2.56111 > 197.22.1.3.80: S
     3204695750:3204695750(0) win 3072
197.22.1.3.80 > 197.22.1.2.56111: R 0:0(0) ack
     3204695751 win 0 (DF)
197.22.1.2.56111 > 197.22.1.3.443: S
     3204695750:3204695750(0) win 3072
197.22.1.3.443 > 197.22.1.2.56111: R 0:0(0) ack
     3204695751 win 0 (DF)
```

TCP port 80 and 443 are allowed from the internal network to the Internet. This is in accordance with the firewall rules (internal users are allowed to browse on the Internet).

```
nmap -P0 -e eth0 -sS -S 197.22.1.2
     -p 1-65535 197.22.1.3
nmap -P0 -e eth0 -sU -S 197.22.1.2
     -p 1-65535 197.22.1.3
icmpush -echo -sp 197.22.1.2 197.22.1.3
```

No packets get through the firewall, which shows that spoofing is disallowed and NAT-ing works properly.

2.1.5 Internal to DMZ

On sniffer1 we start a tcpdump:

```
tcpdump -i eth0 -n host 192.168.1.5
```

and for the spoof tests:

```
tcpdump -i eth0 -n host 197.22.1.225
```

The internal address is not NAT'ed in this case, as it is routed to sniffer1 directly by the firewall.
Then we run the tests from sniffer2:

```
nmap -P0 -p 1-65535 197.22.1.229
nmap -P0 -sU -p 1-65535 197.22.1.229
icmpush -echo 197.22.1.229
nmap -P0 -e eth0 -sS -S 197.22.1.225
     -p 1-65535 197.22.1.229
nmap -P0 -e eth0 -sU -S 197.22.1.225
     -p 1-65535 197.22.1.229
icmpush -echo -sp 197.22.1.225 197.22.1.229
```

No packets seen at sniffer1. Not even TCP port 22 or 80 - these are only allowed to the specific DMZ servers (ssh and web server).

2.1.6 DMZ to Internal

On sniffer2 we start a tcpdump:

```
tcpdump -i eth0 -n host 197.22.1.229
```

and for the spoof tests:

```
tcpdump -i eth0 -n host 192.168.1.1
```

Then we run the tests from sniffer1:

```
nmap -P0 -p 1-65535 192.168.1.5
nmap -P0 -sU -p 1-65535 192.168.1.5
icmpush -echo 192.168.1.5
nmap -P0 -e eth0 -sS -S 192.168.1.1
     -p 1-65535 192.168.1.5
nmap -P0 -e eth0 -sU -S 192.168.1.1
     -p 1-65535 192.168.1.5
icmpush -echo -sp 192.168.1.1 192.168.1.5
```

No packets arrived at sniffer2. Only packets that are part of an established connection to the ssh or web server are allowed from the DMZ according to our firewall rules.

3 Conclusion and Recommendations

The firewall ACL's behave as expected. Spoofing is not possible and only specific necessary services are forwarded.
It is hard for a malicious user to gather information on GIAC's network, because non of the public systems are allowed to reply to ICMP echo requests and closed TCP and UDP ports time out during a port scan. The latter also makes it lengthy to port scan GIAC's systems, which might keep the majority of the "script kiddies" away (security by obscurity).
However, the firewall IP is shown in traceroutes to systems on the DMZ. Adding a rule to the firewall that prohibits ICMP TTL exceeded messages to go out from the Internet interface can eliminate this. The following rule can be used:

```
iptables -A OUTPUT -o eth0
     -p icmp --icmp-type ttl-exceeded -j DROP
```

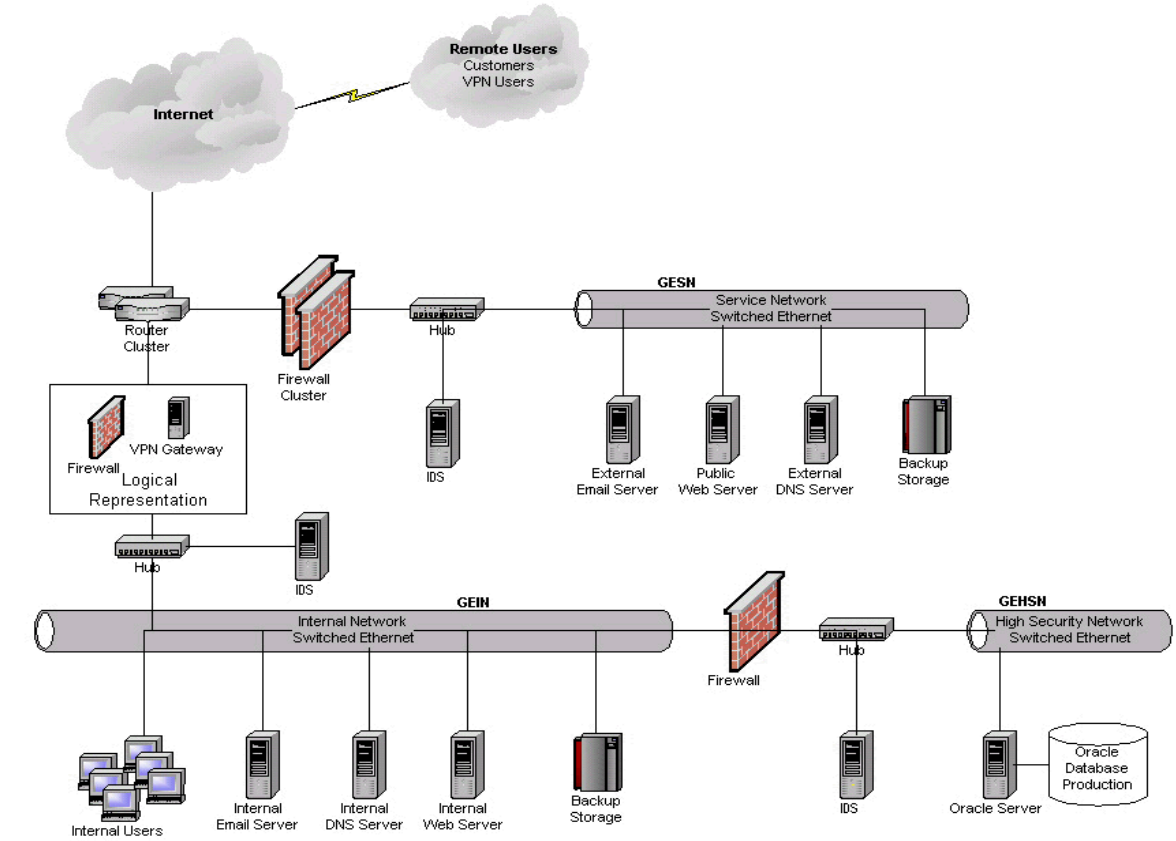
This was only a test of GIAC's firewall. It is recommended to test all other public and internal GIAC systems too with one or more vulnerability scanners to make sure that there are no "weak links in GIAC's security chain".

During the test, we discovered that it takes a long time to connect to TCP port 25 on the mail server. This is caused by the fact that the mail server tries to connect to TCP port 113 (ident/auth) on our source address for authentication purposes. The firewall does not allow this, so the authentication process has to time out before the SMTP connection continues. This is not directly a security issue; it is just some extra information.
A solution might be to disable the authentication on the mail server or to allow connections to TCP port 113 out from the mail server.

Assignment 4 - Design Under Fire

I have chosen Dennis Pickett's (0211) network design for this assignment:
http://www.giac.org/practical/Dennis_Pickett_GCFW.zip

Network Design Schematic



1 Firewall Attack

The external firewalls Dennis Pickett uses are Firewall-1 4.0 build 4094 running on Nokia IP440s with IPSO 3.2.1 (IPSO build 13)
Firewall-1 version 4.0 has several vulnerabilities; I've picked the following three:

1.1 Fragmented Packets DoS

On SecurityFocus the following description can be found[26]:

By sending illegally fragmented packets directly to or routed through Check Point FireWall-1, it is possible to force the firewall to use 100% of available processor time logging these packets. The FireWall-1 rulebase cannot prevent this attack and it is not logged in the firewall logs.

[26] See: SecurityFocus web site: http://online.securityfocus.com/bid/1312
http://online.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=1890

To exploit this vulnerability, the jolt2 exploit, which is posted on SecurityFocus, can be used. This exploit was originally designed to crash Windows hosts (see Microsoft Security Bulletin MS00-029[27]), but it is also effective on Firewall-1.
The exploit code can be downloaded from several places on the Internet (e.g. SecurityFocus). After examination of the source code(!), it can be compiled in with a C-compiler.
The exploit uses ICMP by default, but it can also use UDP:

Usage: `jolt2 [-s source ip] [-p port] destination ip`
ICMP: `jolt2 <firewall-ip>`
UDP: `jolt2 <firewall-ip> -p <udp port>`

To use UDP, we would have to find an open UDP port on the firewall, e.g. by doing a UDP port scan first.
We might also consider covering our tracks by spoofing the source ip (-s option).

We will not be able to verify directly if the firewall crashed (it does not reply to ping), but we can try to connect to the web server behind the firewall. If we are not able to do that anymore, we crashed the firewall.

Checkpoint has a description of a workaround for this vulnerability on their web page[28].

1.2 Valid Username Vulnerability

SecurityFocus description[29]:

Checkpoint Firewall-1 is a popular firewall package available from Checkpoint Software Technologies. A vulnerability exists in Firewall-1 whereby an attacker can determine a valid username by the response given by the firewall to authentication requests (port 259 on the firewall) from a remote client.

Upon connecting to the firewall, the attacker enters a username and password. If the username and password are invalid, the firewall will respond with "<username> not found". If the username is valid, and the password is invalid, the firewall will respond with "Access denied by Firewall-1 authentication".

Upon successfully determining a valid username, a remote attacker could then attempt a brute force or password grinding attack to determine the password for the valid username. If successful, an attacker could then gain access to the firewall based on that user's privileges.

[27] http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/bulletin/ms00-029.asp
[28] http://www.checkpoint.com/techsupport/alerts/ipfrag_dos.html
[29] http://online.securityfocus.com/bid/1890

The only way to exploit this vulnerability is to have a lot of patience and manually find a correct username. This process might be automated though. After finding a username, more patience is needed to brute force the password.

However, exploitation of this vulnerability might only work on the firewall that protects the internal network. The firewall that protects Dennis Pickett's service network only allows access to the firewall management from specific sources. The rules for the internal network firewall are not specified in detail in Dennis Pickett's document.

SecurityFocus has a workaround (non-Checkpoint) on their web site:

Administrators can create a generic account in the user database of FW-1 that will remedy this problem. This account will trigger on all usernames that have not been explicitly been defined in the user database and prevent an attacker from profiling the database.*

This solution is not a vendor fix and was supplied by a bugtraq subscriber. Please see the reference section for the original message.

1.3 SecureRemote Network Information Leak Vulnerability

SecurityFocus description[30]:

SecureRemote is the proprietary VPN infrastructure designed by Check Point Software, and included with some versions of Firewall-1.

A problem with the package allows remote users to gain information about internal networks. Older versions of the package send network topology information to SecureRemote connections prior to authentication, allowing an information gathering attack.

This vulnerability will not give full access directly, but it provides us with some useful network topology information.
The Perl script posted on SecurityFocus can exploit this:

```
sr.pl <firewall-ip>
```

It is not necessary to have a Checkpoint VPN client, but TCP port 256 has to be accessible to get the exploit to work. Since Dennis Picket only allows certain external users to access the VPN on the internal network firewall, we might not be able to exploit the vulnerability.

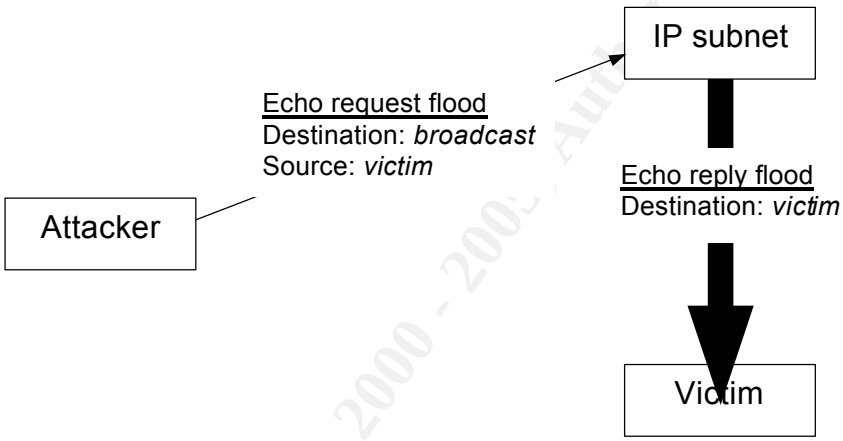
There is a workaround as well as a patch for this vulnerability.
The workaround is to uncheck *respond to unauthenticated topology requests* in the Firewall-1 policy editor.

[30] http://online.securityfocus.com/bid/3058

2 Distributed Denial of Service

For DDoS-ing with we can use the well-known Smurf ICMP Broadcast attack[31].

An attacker will try to send a flood of spoofed ICMP echo requests to the broadcast address of an IP subnet. If the subnet's border router does not prohibit pings to the broadcast address of this IP subnet, most active IP addresses in the subnet will send an echo reply back.
The devious part of the attack is that the attacker will use the IP address of their victim as the spoofed address. This means that the victim will receive a flood of echo replies coming from the IP subnet that allows pings to the broadcast address. This flood will be a multitude of the echo request flood that was send originally.
This looks as follows in a diagram:



We will not be able to use Dennis Pickett's IP subnet as the ping amplifier because pings to the broadcast address are prohibited in his router with the following rule[32]:

```
no ip direct-broadcast
```

However, we can find an IP subnet that has a broadcast address that replies to ping. Using this we can for example attack Dennis Pickett's router or another system in his address space.

The Norwegian company Powertech has on their web site an updated overview of some subnets that can be used as a Smurf amplifier[33].
We copy these subnets into a text file: *subnets*.

[31] See: CERT's web site: http://www.cert.org/advisories/CA-1998-01.html
[32] Dennis Pickett's document page 14
[33] See: Powertech's web site: http://www.powertech.no/smurf/

Next, we execute the exploit that is posted on SecurityFocus[34] from each cable modem/DSL system that we have available, using the subnet text file:
Usage:

```
smurf <target> <subnet file> <nr of packets>
      <packet delay> <packet size>
```

Where:

```
target       = victim
subnet file  = file to read broadcast addresses from
num packets  = number of packets to send (0 = flood)
packet delay = wait between each packet (in ms)
packet size  = size of packet (< 1024)
```

If we choose Dennis Pickett's router, the command would be:

```
Smurf <router> subnets 0 1 512
```

In this way we can send a huge number of packets per second. This might actually also congest the amplifier subnet's network.

It is very hard for the victim to prevent these kind of attacks. Even though the victim's systems do not reply to ping, the victim's Internet connection might get congested.
If an attack like this occurs, the victim should contact the system administrator(s) of the IP subnet that is used as an amplifier and educate them on how to prohibit echo request to their broadcast address.

3 Internal Attack

I've decided to try to attack the web server on the service network. This is an Apache web server running on Solaris. The web server version is not stated in Dennis Pickett's document.
Only HTTP and HTTPS are allowed to the web server[35].

When installing Apache, mostly PHP is installed too by default. Therefore I've chosen the following vulnerability (SecurityFocus description[36]):

PHP Post File Upload Buffer Overflow
PHP is a widely deployed scripting language, designed for web based development and CGI programming.

PHP does not perform proper bounds checking on in functions related to Form-based File Uploads in HTML (RFC1867). Specifically, this problem occurs in the functions which are used to decode MIME encoded files. As a

[34] See: The SecurityFocus web site: http://downloads.securityfocus.com/vulnerabilities/exploits/smurf.c
[35] Dennis Pickett's document page 21
[36] See: The SecurityFocus web site: http://online.securityfocus.com/bid/4183

result, it may be possible to overrun the buffer used for the vulnerable functions to cause arbitrary attacker-supplied instructions to be executed.

PHP is invoked through webservers remotely. It may be possible for remote attackers to execute this vulnerability to gain access to target systems. A vulnerable PHP interpreter module is available for Apache servers that is often enabled by default.

If we assume that PHP is installed on the web server, we are most likely to succeed in our attack.
An easy way to find out whether PHP is running is to open a telnet connection to TCP port 80 on the web server and issue:

```
HEAD / HTTP/1.0
```

In most of the cases, the Apache web server displays PHP installation information in the returned banner.

Author's note: At the time I wrote this document, the PHP vulnerability was very new. It was therefore very hard to find an exploit for it.

A patch for this vulnerability can be downloaded form www.php.net.

APPENDIX A - Netfilter/iptables script

```
#!/bin/bash
# This firewall script is for GIAC Enterprises
#
# 2002 - Roel Schouten
#

#------------------------------------------------------------
# Initialization
#------------------------------------------------------------

echo "GIAC Firewall started at `date`" > /var/log/messages

echo -n "Initialization                          :"
# Location of the iptables binary
IPTABLES="/sbin/iptables"

# The used network interfaces
LOCAL_INT="lo"                # Loopback Interface
INTERNET_INT="eth0"           # External Interface to Internet
DMZ_INT="eth1"                # DMZ Interface to servers
INTERNAL_INT="eth2"           # Internal Interface to LAN
IPSEC_INT="ipsec0"            # IPsec interface

# Define and automatically obtain the IP-addresses of the network interfaces:
INTERNET_IP="`ifconfig $INTERNET_INT | grep \"inet addr\" | cut -f 2 -d \":\" | cut -f 1 -d \"
\"`"
DMZ_IP="`ifconfig $DMZ_INT | grep \"inet addr\" | cut -f 2 -d \":\" | cut -f 1 -d \" \"`"
INTERNAL_IP="`ifconfig $INTERNAL_INT | grep \"inet addr\" | cut -f 2 -d \":\" | cut -f 1 -d \"
\"`"

# Define the local loop back IP
LOCAL_IP="127.0.0.0"
echo "Done"

# Define GIAC systems
ROUTER="197.22.1.1"
DNSSERVER="ns.giac-isp.net"
WEBSERVER="197.22.1.226"
MAILSERVER="197.22.1.227"
SSHSERVER="197.22.1.228"
GIACSUBNET="192.168.1.0/24"
FORTUNEDB="192.168.1.3"
ADMINMACHINE="192.168.1.4"
PARTNERFW="fw.giac-partner.net"
PARTNERSUBNET="10.1.1.0/24"


#------------------------------------------------------------------------------
# Kernel Settings
#------------------------------------------------------------------------------

echo -n "Applying kernel settings                :"
# Enable forwarding
echo "1" >/proc/sys/net/ipv4/ip_forward

# Enable syn-cookies to prevent syn-flooding attacks
echo "1" >/proc/sys/net/ipv4/tcp_syncookies

# Disable ICMP echo-request to broadcast addresses to prevent Smurf attack
echo "1" >/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Shut off source-routing and enable IP spoof detection
# This has to be done for all network interfaces
for f in /proc/sys/net/ipv4/conf/*; do
   # Drop all source-routed packets
   echo "0" >$f/accept_source_route

   # Enable source-address verification (anti spoofing).
   # The value 2 means use Ingress filtering as per RFC 1812.
   echo "2" >$f/rp_filter
```

```
done
echo "Done"


#------------------------------------------------------------------------
# Flushing Existing Connections
#------------------------------------------------------------------------

echo -n "Flushing existing connections         :"
$IPTABLES -t filter -F
$IPTABLES -t nat -F
echo "Done"


#------------------------------------------------------------------------
# Chains Definition and Initialization
#------------------------------------------------------------------------

echo -n "Chains definition and init            :"
# Default chains definition
$IPTABLES -P INPUT DROP       # Drop all packets to input
$IPTABLES -P FORWARD DROP     # Don't forward anything
$IPTABLES -P OUTPUT ACCEPT    # Accept all packets from output

# Flush the default chains
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
$IPTABLES -F OUTPUT
$IPTABLES -Z INPUT
$IPTABLES -Z FORWARD
$IPTABLES -Z OUTPUT

# User chains definition
$IPTABLES -N local
$IPTABLES -N internal
$IPTABLES -N internet
$IPTABLES -N dmz
$IPTABLES -N internet_to_dmz
$IPTABLES -N dmz_to_internet
$IPTABLES -N internal_to_dmz
$IPTABLES -N dmz_to_internal
$IPTABLES -N internal_to_internet
$IPTABLES -N internet_to_internal
$IPTABLES -N vpn_in
$IPTABLES -N vpn_out

# Flush the GIAC user chains
$IPTABLES -F local
$IPTABLES -F internal
$IPTABLES -F internet
$IPTABLES -F dmz
$IPTABLES -F internet_to_dmz
$IPTABLES -F dmz_to_internet
$IPTABLES -F internal_to_dmz
$IPTABLES -F dmz_to_internal
$IPTABLES -F internal_to_internet
$IPTABLES -F internet_to_internal
$IPTABLES -F vpn_in
$IPTABLES -F vpn_out
echo "Done"


#------------------------------------------------------------------------
# NAT
#------------------------------------------------------------------------

echo -n "Setting up NAT                        :"
# Flush NAT-chain POSTROUTING
$IPTABLES -t nat -F POSTROUTING

# POSTROUTING and SNAT for all packets that go out from the Internal network to the Internet
$IPTABLES -t nat -A POSTROUTING -s $GIACSUBNET -o $INTERNET_INT -j SNAT --to-source
```

```
$INTERNET_IP
echo "Done"

#------------------------------------------------------------------------
# Interface Rules
#------------------------------------------------------------------------

echo "Setting up Interface rules"

# *** Local loop back ***
echo -n "Local interface                        :"
# Allow all connections to the local interface, if the source IP is the local loop back to
# make sure that Linux can talk to itself
$IPTABLES -A local -m state --state NEW,ESTABLISHED,RELATED -i $LOCAL_INT -j ACCEPT
echo "Done"


# *** Internal interface ***
echo -n "Internal interface                     :"
# Accept ICMP pings for testing purposes from the Internal network
$IPTABLES -A internal -s $GIACSUBNET -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A internal -s $GIACSUBNET -p icmp --icmp-type echo-reply -j ACCEPT

# Log everything else
$IPTABLES -A internal -j LOG --log-prefix "FW-LOG Internal interface:"

#Drop everything else
$IPTABLES -A internal -j DROP
echo "Done"


# *** Internet interface ***
echo -n "Internet interface                     :"
# All traffic to the Internet interface is logged
$IPTABLES -A internet -j LOG --log-prefix "FW-LOG Internet interface:"

# Allow VPN from the Partner with IKE and ESP
$IPTABLES -A internet -p udp -s $PARTNERFW --sport 500 --dport 500 -j ACCEPT
$IPTABLES -A internet -p 50 -j ACCEPT

# Drop everything else. No other connections need to be made to the firewall directly from the
Internet
$IPTABLES -A internet -j DROP
echo "Done"


# *** DMZ interface ***
echo -n "DMZ interface                          :"
# Accept ICMP pings for testing purposes from the DMZ
$IPTABLES -A dmz -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A dmz -p icmp --icmp-type echo-reply -j ACCEPT

# Log everything else
$IPTABLES -A dmz -j LOG --log-prefix "FW-LOG DMZ:"

#Drop everything else
$IPTABLES -A dmz -j DROP
echo "Done"


#------------------------------------------------------------------------
# Internet to DMZ
#------------------------------------------------------------------------

echo -n "Setting up Internet to DMZ rules      :"
# All traffic from the Internet to the DMZ is logged
$IPTABLES -A internet_to_dmz -j LOG --log-prefix "FW-LOG Internet to DMZ:"

# Forwarding to the various servers on the DMZ
# Web server HTTP
$IPTABLES -A internet_to_dmz -d $WEBSERVER -p tcp --dport http -j ACCEPT
```

```
# SSH server
$IPTABLES -A internet_to_dmz -d $SSHSERVER -p tcp --dport ssh -j ACCEPT

# Mail server SMTP and SSH (for tunneling POP3)
$IPTABLES -A internet_to_dmz -d $MAILSERVER -p tcp --dport smtp -j ACCEPT
$IPTABLES -A internet_to_dmz -d $MAILSERVER -p tcp --dport ssh -j ACCEPT

# Stateful check of established connections from the mail server
$IPTABLES -A internet_to_dmz -d $MAILSERVER -m state --state ESTABLISHED -j ACCEPT

#Drop everything else
$IPTABLES -A internet_to_dmz -j DROP
echo "Done"


#--------------------------------------------------------------------------
# DMZ to Internet
#--------------------------------------------------------------------------

echo -n "Setting up DMZ to Internet rules      :"
# All traffic from the DMZ to the Internet is logged
$IPTABLES -A dmz_to_internet -j LOG --log-prefix "FW-LOG DMZ to Internet:"

# Only mail server SMTP and DNS out
$IPTABLES -A dmz_to_internet -s $MAILSERVER -p tcp --dport smtp -j ACCEPT
$IPTABLES -A dmz_to_internet -s $MAILSERVER -d $DNSSERVER -p udp --dport 53 -j ACCEPT

# Stateful check of established connections
$IPTABLES -A dmz_to_internet -m state --state ESTABLISHED -j ACCEPT

# Drop everything else
$IPTABLES -A dmz_to_internet -j DROP
echo "Done"

#--------------------------------------------------------------------------
# Internal Network to DMZ
#--------------------------------------------------------------------------

echo -n "Setting up DMZ to Internal rules    :"
# SSH server access for fortune update
$IPTABLES -A internal_to_dmz -d $SSHSERVER -p tcp --dport ssh -j ACCEPT

# Mail server SMTP and SSH (for tunneling POP3)
$IPTABLES -A internal_to_dmz -d $MAILSERVER -p tcp --dport smtp -j ACCEPT
$IPTABLES -A internal_to_dmz -d $MAILSERVER -p tcp --dport ssh -j ACCEPT

# Web server HTTP
$IPTABLES -A internal_to_dmz -d $WEBSERVER -p tcp --dport http -j ACCEPT

# Log everything else
$IPTABLES -A internal_to_dmz -j LOG --log-prefix "FW-LOG Internal to DMZ:"

#Drop everything else
$IPTABLES -A internal_to_dmz -j DROP
echo "Done"


#--------------------------------------------------------------------------
# DMZ to Internal Network
#--------------------------------------------------------------------------

echo -n "Setting up DMZ to Internal rules      :"
# Stateful check of established connections
$IPTABLES -A dmz_to_internal -m state --state ESTABLISHED -j ACCEPT

# Log everything else
$IPTABLES -A dmz_to_internal -j LOG --log-prefix "FW-LOG DMZ to internal:"

#Drop everything else
$IPTABLES -A dmz_to_internal -j DROP
echo "Done"
```

```
#-----------------------------------------------------------------------------
# Internal Network to Internet
#-----------------------------------------------------------------------------

echo -n "Setting up Internal to Internet rules :"

# HTTP and HTTPS out
$IPTABLES -A internal_to_internet -p tcp --dport http -j ACCEPT
$IPTABLES -A internal_to_internet -p tcp --dport https -j ACCEPT

# DNS out to the external DNS server
$IPTABLES -A internal_to_internet -d $DNSSERVER -p udp --dport 53 -j ACCEPT

# FTP out from the administrator machine
$IPTABLES -A internal_to_internet -s $ADMINMACHINE -p tcp --dport ftp -j ACCEPT

# Telnet access to the Border router from the administrator machine
$IPTABLES -A internal_to_internet -s $ADMINMACHINE -d $ROUTER -p tcp --dport telnet -j ACCEPT

# Stateful check of Partner connections to Fortune database (no connection initiations are
necessary/allowed from the Fortune database to the partner)
$IPTABLES -A vpn_out -m state --state ESTABLISHED,RELATED -j ACCEPT

# Log everything else
$IPTABLES -A internal_to_internet -j LOG --log-prefix "FW-LOG Internal to internet:"

# Drop everything else.
$IPTABLES -A internal_to_internet -j DROP
echo "Done"


#-----------------------------------------------------------------------------
# Internet to Internal network
#-----------------------------------------------------------------------------

echo -n "Setting up Internet to Internal rules :"
# Packets from the Internet to Internal network must have an establishment (stateful check).
$IPTABLES -A internet_to_internal -m state --state ESTABLISHED,RELATED -j ACCEPT

# Log everything else
$IPTABLES -A internet_to_internal -j LOG --log-prefix "FW-LOG Internet to internal:"

# VPN from the Partner to Fortune DB
$IPTABLES -A vpn_in -s $PARTNERSUBNET -d $FORTUNEDB -j ACCEPT

# Drop everything else.
$IPTABLES -A internet_to_internal -j DROP
echo "Done"


#-----------------------------------------------------------------------------
# Activate chains
#-----------------------------------------------------------------------------

echo -n "Activating user chains               :"
# Activate the user chains
$IPTABLES -A INPUT -i $LOCAL_INT  -j local
$IPTABLES -A INPUT -i $INTERNAL_INT -j internal
$IPTABLES -A INPUT -i $INTERNET_INT -j internet
$IPTABLES -A INPUT -i $DMZ_INT -j dmz
$IPTABLES -A FORWARD -i $INTERNET_INT -o $DMZ_INT -j internet_to_dmz
$IPTABLES -A FORWARD -i $DMZ_INT -o $INTERNET_INT -j dmz_to_internet
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $DMZ_INT -j internal_to_dmz
$IPTABLES -A FORWARD -i $DMZ_INT -o $INTERNAL_INT -j dmz_to_internal
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $INTERNET_INT -j internal_to_internet
$IPTABLES -A FORWARD -i $INTERNET_INT -o $INTERNAL_INT -j internet_to_internal
$IPTABLES -A FORWARD -i $IPSEC_INT -o $INTERNAL_INT -j vpn_in
$IPTABLES -A FORWARD -i $INTERNAL_INT -o $IPSEC_INT -j vpn_out
echo "Done"

echo "All done"
```

References

http://www.iana.org/
http://www.ietf.org/
http://www.counterpane.com/
http://www.redhat.com/
http://www.netfilter.org/
http://www.freeswan.org/
http://www.apache.org/
http://www.snort.org/
http://www.sendmail.org/
http://www.eudora.com/
http://www.openssh.com/portable.html
http://www.zonealarm.com/
http://www.mcafee.com/
http://www.securityfocus.com/
http://bmrc.berkeley.edu/
http://www.ssh.com/
http://www.cert.org/
http://www.cisco.com/
http://www.netfilter.org/
http://www.insecure.org/
http://michael.toren.net/
http://www.packetstormsecurity.org/
http://www.microsoft.com
http://www.checkpoint.com
http://www.powertech.no/